

JAVA PROGRAMMING

ICS4M

COMPUTER AND INFORMATION SCIENCE



John Ketelaars
Consultant
Computer Science

Acknowledgments

In unit 3 I have used the example in the Appendix of **Hume and Stephenson's** *Introduction to Programming in Java* as a model Case study to follow. I found it an excellent example to consider.

In the sections on Ergonomics, Career Research and Information Technology and the Community, I have adapted the information developed by by **Yee-Min Cha** of John Fraser SS, Peel DSB in her unit *Social Impact and Consequences*.

I thank the TDSB and Robert Wager, the district wide coordinator of Technology for having given me the opportunity to write this profile.

Preface

I have taken the Ministry sponsored Course Profile and changed the order. I found that it would be useful to introduce recursion at the start of the course. Recursion is then the theme of the first unit. Records and Binary files can best be treated using object oriented techniques, so that is the theme of unit 2. Both units are rather long, but they hold interesting topics and replace the first three units of the Ministry sponsored Profile both in time and topics.

As in the Course Profile for grade 11, I used Java as the chosen language. With slight modifications the course can be adapted to a different object oriented language such as C++ or Turing

Units:

Unit 1	Recursions and Tables-----	21	days
Unit 2	Objects and Data Structures-----	30	days
Unit 3	Managing Software Projects-----	16	days
Unit 4	Applying Project Management & Software Development Skills----	21	days

TOTAL 88 days

Table of Contents

Unit 1 Recursions and Tables -----	4
The importance of Ergonomics-----	5
Simple Recursions-----	7
Arrays and Text Files-----	11
Recursions Revisited-----	13
Career Research-----	18
Unit 2 Objects and Data Structures -----	21
Applets and Objects-----	22
Records and Binary Files-----	26
Sorting and Searching Techniques-----	30
Linked Lists-----	32
Unit 3 Managing Software Projects -----	34
The Software design life cycle-----	35
Problem Definition and Analysis-----	37
Program Design-----	39
Problem Implementation-----	41
Problem Testing-----	43
Unit 4 Applying Project Management & Software Development Skills -----	44
Information Technology and the Community-----	45
Defining and analyzing the problem-----	47
Making a plan and defining the roles-----	49
Creating and testing a solution-----	51
Documenting the solution-----	52
Appendix -----	53
Accommodations, Resources, Teaching and learning strategies, Assessment and Evaluation-----	53
Skills inventory-----	55
Paper and Pencil Sorting Examples-----	58

Unit 1: Recursions and Tables

Time: 21 days

Unit Description

In this unit, students review and extend their knowledge of Arrays, Tables and Text Files. They will be introduced to recursion and explore both its linear and non-linear manifestations.

Students also review and reinforce the principles of ergonomics and consider how it applies to themselves and their environment. They explore career opportunities in computing and information science related fields.

Unit Overview Chart

Cluster	Expectations	Assessment	Focus	# of days
1	IC2.04	K/U, C	The importance of ergonomics	2
2	TFV.03 TF1.04 TF2.03 SP2.06 SP2.07	K/U, A	Simple Recursions	6
3	TFV.02 TF3.01 SPV.04 SP1.05 SP2.03	K/U, A	Arrays and Text Files	5
4	TFV.03 TF2.03 SP2.07 SPV.02 SP1.08	T/I, A	Recursions Revisited	6
5	ICV.01,ICV.03, ICV04, IC3.01 IC3.02	K/U, C	Career research	2

K/U = Knowledge/Understanding
T/I = Thinking/Inquiry

C = Communication
A = Application

1.1 The importance of Ergonomics

2 days

Description

On the first day Students are introduced to the lab and are given the outline and expectations of the course. Students will consider the safety of the lab and how to prevent accidents and the unwanted side effects of sitting in front of a monitor for extended periods of time.

Strand(s) & Learning Expectations:

Impact and Consequences

IC2.04 Use appropriate strategies to avoid potential health and safety problems associated with computer use, such as musculo-skeletal disorders and eye strain.

Prior Knowledge & Skills

Students should be familiar with the use of the computer, and since the ICS3M course is prerequisite to this course, that is invariably the case.

Planning Notes

Make a list of some of the web sites available for students to review the good and bad points of the computer lab students will be using.

Teaching/Learning Strategies

- Explain the rules regarding the use of the lab, the network, and the software.
- There are certain protocols to be followed at the start of the period and at the end of the period.
- Students should know what to do, when something is found broken or missing at their workstation.
- There are rules to be followed regarding the use of the printer
- The use of floppy disks may be restricted or banned

Where to keep your work

- Students may be assigned space on the school's server
- Students may be required to have their own e-mail address

Uses of Spreadsheets, Databases, Word processors.... and Notebooks

- Depending on the environment available, students may be encouraged to develop a glossary of terms, a list of specific functions, methods, useful examples and record these on the appropriate application.
- Students should be encouraged to keep a notebook where they record their due dates and keep (sample) printouts of their work.

Resources:

- **IBM's Healthy Computing site**

A web site devoted to helping you use your personal computer comfortably, providing extensive ergonomic and environmental information at one location.

<http://www.pc.ibm.com/ww/healthycomputing/>

- **Safe Office Practice**

This web site is designed to help you avoid or reduce health problems arising from computer use. Most of these problems can be helped by making straightforward changes to the layout of your desk and computer. Symptoms typically escalate very quickly from persistent discomfort to chronic pain which can end your career and reduce your quality of life in wide-ranging ways.

<http://www.openerg.com/dse/index.html>

- **Workstation ergonomics: combating computer fatigue**

Workstation ergonomic tips that will help along with effective exercises that can be integrated into a workday alleviating & avoiding these symptoms (eyestrain and muscle tension) of computer fatigue.

http://oror.essortment.com/workstationergo_rdum.htm

- **School Ergonomics Programs: Guidelines for Parents**

With the number of computers in classrooms increasing every day, many schools are beginning to institute ergonomics programs to show students, teachers, and parents how to reduce the risks of computer-related injuries. What follows is part of such an ergonomics program.

<http://ergo.human.cornell.edu/MBergo/intro.html>

Exercises

- 1.1.1 Create a database spreadsheet or word processor document where you will keep vocabulary or glossary of terms learned in this course. Consider allocating space not only for definitions of terms but also the context, which might include examples.
- 1.1.2 Individually or in groups of two, review the literature available and list all the details you can find that form part of a correct posture in front of the computer. Then decide on the three most common failings.
- 1.1.3 Individually or in groups of two, review the literature available and make a list of how an ideal workstation should look like. Then consider your school lab, and list the discrepancies.
- 1.1.4 In groups of 4 or 5 brainstorm in search of practical solutions for improving the lab's setting or your own habits of work.
- 1.1.5 Individually or in groups of two, review the literature available and list the three most common ailments related to working on a computer, and find how to prevent or deal with these ailments.

1.2 Simple Recursions

6 days

Description

The essential concept of recursion is a method that calls itself. It is therefore a form of repetition. Students will expand their knowledge of repetition, by converting various standard problems, frequently solved by a **for** or **while** loop. Students should be familiar with methods. In many initial examples this merely becomes a routine translation, but this is important for gaining confidence and understanding of the concept. In later examples it becomes challenging to view a problem in such a way that it can be expressed in a recursive fashion.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.03 analyze a number of programming paradigms;

TF1.04 evaluate the efficiency of different algorithms and their applicability to solving the same programming problem;

TF2.03 explain how recursion can be used to solve specific kinds of computing problems.

Skills and Processes

SP2.06 use recursion in a simple program;

SP2.07 compare the effectiveness of several algorithms for solving the same problem

Prior Knowledge & Skills

Student should be familiar with:

- Simple variables
- Strings and their functions
- Mathematical functions
- Counted and conditional loops,
- methods (procedures and functions)

Planning Notes

This is the first programming topic in this course. Recursion is something students have not seen before, and this will set the tone of the course: There is a lot to learn in this course.

At the same time, the introductory problems offer a quick review of some of the topics from grade 11. Be prepared for a need to review some grade 11 material.

There are a few problems that demand a careful analysis of a problem. The **Tower of Hanoi** is one such problem. Even though the solution is short, developing the solution takes a lot of careful analysis. Review the literature.

Teaching/Learning Strategies

- Review methods, especially those that return a variable.
- Review the basic parts of a loop, including the exit condition.
- Compare the simple recursions to loops.
- Use students' knowledge of mathematical sequences and series to highlight the difference between a function defined in terms of "x" and one defined in terms of its predecessor.
- Discuss the stack and its function
- Discuss infinite loops and infinite recursions and stack overflow error messages.
- Illustrate (with the fibonacci function, for example) that some repetition problems are best solved using loops.

Assessment & Evaluation of Student Achievement

- Through class discussion, informal review (at the start of each class, for example) go over the vocabulary of recursion and related programming techniques.
- Give students a test or assignment to illustrate their skill in solving a given problem using recursive techniques

Resources (book)

John Carter. *Problem solving in Pascal* 1989. Don Mills: Addison Wesley , Chapter 12
Still a classic source for good lesson plans.

(web sites)

Recursion Programming

<http://www11.brinkster.com/erwnerve/recursion.htm>

Recursion (Lecture 3) by Prof. Robert C. Holte, Ottawa University

<http://www.csi.uottawa.ca/~holte/T26/lecture3.html>

RECURSION by Thomas Jenkins, New Mexico State University

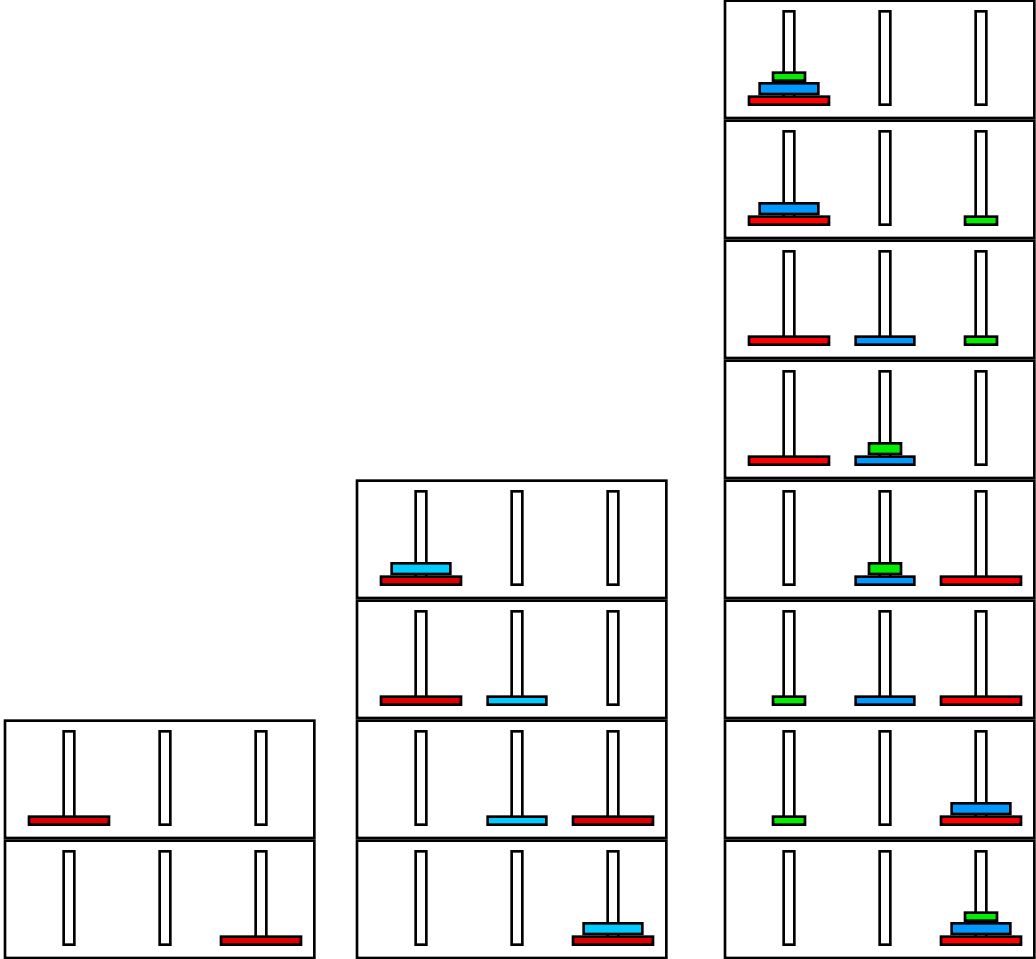
<http://et.nmsu.edu/~etti/winter98/computers/jenkins/jenkins.html>

Exercises

- 1.2.1 Create a method that will return the factorial for any integer, zero or greater: **long factorial (long n)**. Use a counted loop. Then write a program that will test the method by letting you enter any integer. Your testing program should repeat asking for input, until a negative number is entered. (Note that only numbers ≤ 20 will give you valid answers)
- 1.2.2 Use recursion to write the same **long factorial (long n)** and test it with the same program.
- 1.2.3 Create a method that will return a string of characters: **String repeat(char c, int n)** should return return n characters c. For example, `repeat('x',10)` would return; "xxxxxxxxxx"
Use a for loop first, and test it.
- 1.2.4 Create the repeat method above using recursion and test with the same program.
- 1.2.5 Create the method **String expand(String word)** that will insert minus signs between letters. For example, `expand("automobile")` to: "a-u-t-o-m-o-b-i-l-e"
Single letter words are clearly not affected by `expand`. Use recursion.
- 1.2.6 Modify the method above by creating a method that will return the word backwards as well. For example, `reverse("car")` would return: "r-a-c". Use recursion.

- 1.2.7 Create a method **String compact(String line)**, that will eliminate all the spaces from line. For example, **compact("this is a test ")** would return: **"thisisatest"**. Use recursion
- 1.2.8 Create a method **int count(String line, char c)** that will count the number of occurrences of the character c. For example **count("this is a test ", 's')** will return 3, and **count("this is a test ", 'x')** will return 0. Use recursion.
- 1.2.9 The greatest common divisor of (a and b) is the same as the greatest common divisor of (b and c) where c is the remainder, when a is divided by b. Since the remainder of any division is always smaller than the original divisor (i.e. $c < b$), by repeating the argument, the second term will eventually become so small it is zero:
 $\text{gcd}(a,b) = \text{gcd}(b,c) = \dots \text{gcd}(d,0)$ where the greatest common divisor is d.
 Write **int gcd(int a, int b)** that will find the greatest common divisor of a and b using recursion.
- 1.2.10 The cube root of a number can be found based on the observation, that:
 if t is an approximation of the cube root of a, then $\frac{\frac{a}{t^2} + 2t}{3}$ is a better approximation.
 Create a method **double betterCubeRoot(double a, double t)**. that will find the cube root of a accurate enough so that the difference between t^3 and a is less than .0001. Use recursion.
 Then write the method **double cubeRoot(double a)** that makes use of the method with 1 the initial value for t. Write a program that will test values both positive and negative.
- 1.2.11 The 5th root of any number can similarly be found: If t is an approximation of the 5th root of a, then $\frac{\frac{a}{t^4} + 4t}{5}$ is a better approximation. Use recursion to create a method for finding the 5th root: **double fifthRoot(double a)**.
- 1.2.12 An equation of the form $ax^3 + bx^2 + cx + d = 0$ always has at least one solution. And if t is one approximation, then $\frac{2at^3 + bt^2 - d}{3at^2 + 2bt + c}$ is a better one.
 Use recursion to create the method **realRoot()** for finding a root for the equation $ax^3 + bx^2 + cx + d = 0$.

1.2.13 Write a program that will print out the solution of how to move the disk in the "Towers of Hanoi" problem from one pole to another, using recursion, given the number of disks to start: e.g.:
 Here are examples of the 1,2 and 3 disk case.
 A very good treatment is given in Carter's book.



1.3 Arrays and Text files

5 days

Description

In this section students review and expand their knowledge of arrays, tables and text files. they will be introduced to some simple techniques of manipulating and managing data, including the linear search. Most of the data will come from the web, and in particular from Statistics Canada.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.02 explain data structures and their processing algorithms;

TF3.01 explain the role of a network in accessing computer software resources

Skills and Processes

SPV.04 identify on-line and off-line resource materials;

SP1.05 select appropriate data structures (e.g., arrays, records, arrays of records) for use in projects

SP2.03 use arrays, records, and arrays of records in different project settings

Planning Notes

Review the suggested resources for students to use.

If the lab is not networked, it may be useful to make available several of the data in other forms, such as in text or spreadsheet format.

Teaching/Learning Strategies

Encourage students to break up their problems into methods (procedures and functions), each with a short explanation of its function (internal documentation)

If you are using "Ready", it will be extremely useful to use `hsa.Textconsole` for the screen output, It will allow for an indefinite number of output lines.

This section is useful to concentrate on formatted output of the data; printed lists should have headings;

The most common logical errors are "going beyond the limits of arrays"

Trap errors by introducing temporary print statements that allows students to track counters.

Resources

(books)

- Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4 chapter 13, **Arrays**
- Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4 pp138-141, **text files**
- Vincent Lo. *Learning Object Oriented Programming with Java*, 2002. ISBN 0-9695844-2-4, Unit 6: **Arrays**.

(web sites)

- Introduction to Computer Science using Java
- chortle.ccsu.edu/cs151/cs151java.html
- Java Tutorial (IO streams)
- <http://java.sun.com/books/tutorial/essential/>

Exercise problems:

- 1.3.1 Write a program that lets you enter a number from 1 to 7 and that returns the name of the day of the week. Your program should continue asking for week numbers until you enter an invalid integer, at which time it stops. Use an array which you initialize with the days of the week
- 1.3.2 Modify the program above to convert numbers to names of months.
- 1.3.3 Write a program that lets you enter the day and month number, and that will calculate the day of the year. Initialize an array with the total number of days in each of the 12 months of 2002. If for example you enter the numbers 5 and 2, representing February 5, your program should return the number 36.
- 1.3.4 See: <http://www.statcan.ca/english/Pgdb/demo02.htm> and use it to create a text file with 6 columns. Space the data in such a way, that provinces start at position 0, the 1997, 1998, 1999, 2000 and 2001 data respectively at positions 30, 40, 50, 60 and 70. Remove the commas from the numbers.
Write a program that will read the data into 2 arrays: **String province[13]** and **double year[13][5]**. Then print the data on the screen.
- 1.3.3 Write a program that lets you update the text file in 1.3.4 by adding the 1996 data to it and saving the new table in a new text file. You may obtain the 1996 data from: <http://www12.statcan.ca/english/census01/products/standard/popdwell/Table-PR.cfm>
- 1.3.5 See: <http://www.statcan.ca/english/Pgdb/People/Population/demo05.htm> and similar to the previous problem, prepare a text file and write a program that will let you enter a single letter and then print the data for all cities starting with that letter.
- 1.3.6 Let **diff** be the increase in population between 1997 and 2001 (in case of decrease, **diff** would be a negative number). Let **pop** be the population in 1997. Define factor as: $\text{factor} = (1 + \text{diff}/\text{pop})^{1/4}$
Then one can calculate the expected population by repeatedly multiplying the 1997 population figure by **factor**.
For example, the estimated population in 2007 is: $\text{pop} \times \text{factor}^{10}$
Write a program that will let you enter a year > 2000 and that will print out a list of all 25 metropolitan areas and the expected population at that time.
- 1.3.7 Modify the program above to print only the results of the top 10 cities by population. Use the bubble sort method.

- 1.3.8 Modify the program to print four columns of 10: the first two columns are the top ten cities by population in 2001 with their population at that time, and the second set of two columns are the top ten cities by population in the requested year with their population at that time.

1.4 Recursions revisited

6 days

Description

The non-linear Recursions are characterized by methods that call themselves in more than one place. This technique allows for searches in multiple directions, or allows for algorithms to be effective over a wide range of data. Controlling when the recursion should stop becomes much more challenging. Students will be extending their error trapping skills. Programming assignments will make use of text files, and will include the simulation of solving mazes, and filling the area of an irregular shape.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.03 analyze a number of programming paradigms;

TF2.03 explain how recursion can be used to solve specific kinds of computing problems.

Skills and Processes

SP2.07 compare the effectiveness of several algorithms for solving the same problem

SPV.02 effectively use software development and diagnostic tools;

SP1.08 use a problem-solving protocol to troubleshoot computer programs.

Prior Knowledge & Skills

Students must be familiar with text files and linear recursion techniques.

They must have some problem solving and troubleshooting skills.

Planning Notes

Many of these problems are based on previously prepared data. Prepare therefore several variations of text files to illustrate your programming examples.

Teaching/Learning Strategies

Go slow teaching this topic.

Encourage students to separately test each part of their program.

Encourage them to modify the text file using Notepad in the process of troubleshooting

Resources

(Web sites)

Recursion by

<http://erwnerve.tripod.com/prog/recursion/index.htm>

Examples:

```
//linear example
public static void test (int level)
{
    cs.println ("this is level " + level);
    if (level < 10)
        test (level + 1);
}

//variation:
public static void test (int level)
{
    cs.println ("entering level " + level);
    if (level < 10)
        test (level + 1);
    cs.println ("leaving level " + level);
}

//nonlinear (it splits in two at each new level,
// and prints not just 1+1+1+1=4 times, but 2+4+8+16=30 times!
static int count=0;
public static void test2 (int level)
{
    cs.println ("I am at the top of " + level+" for count "+(++count));
    if (level < 4)
        test2 (level + 1);
    cs.println ("I am at the bottom of " + level+" for count "+(++count));
    if (level < 4)
        test2 (level + 1);
}
```

Exercises:

1.4.1 Searching in two dimensions: Find the best route:

```
12 10 33 21
15 20 14 24
18 15 52 5
22 21 30 41
```

Imagine you are at the bottom left corner, and want to go to the top right of this array.

You may only move west or north, and therefore must make 6 moves altogether. At each point you must pay a toll: the indicated amount of dollars. Write a program that will find the minimum amount of money you can spend.

```

// The "C12p1401" class.
import java.awt.*;
import hsa.Console;

public class C12p1401
{
    static Console cs = new Console ();
    static int M [] [] = {
        {12, 10, 33, 21},
        {15, 20, 14, 24},
        {18, 15, 52, 5},
        {22, 21, 30, 41}};
    static int min;

    public static void main (String args [])
    {
        printArray ();
        min = 10000;
        int sum = 0;
        findMin (0, 0, sum);
        cs.println ("the minimum path = " + min);
    }

    // printing the array
    static void printArray ()
    {
        for (int i = 0 ; i < 4 ; i++)
        {
            for (int j = 0 ; j < 4 ; j++)
                cs.print (M [i] [j],5);
            cs.println ();
        } //end i-loop
    } //end printArray

    // You are at (a,b) and add the number there.
    // Then you proceed in one of two possible directions
    // It is important that min is a GLOBAL variable
    static void findMin (int a, int b, int sum)
    {
        sum += M [a] [b];
        if (a < 3)
            findMin (a + 1, b, sum);
        if (b < 3)
            findMin (a, b + 1, sum);
        if (a + b == 6 && min > sum)
            min = sum;
    }
} // C12p1401 class

```

- 1.4.2 Modify the method above, so that it is possible to go diagonally North-West as well, however, by going "cross-country" the toll will then double, and so, to move from position (2,2) where you must pay \$15, to (3,3) where normally you must pay \$14, you must instead pay \$28.

Use the method below to will find the length of the path, knowing that the path starts in the upper left hand corner, and that it will always either go right or down. The values of x and y are passed along down the stack: they are represent the location in the rectangle.

```
static String maze [];  
static int mazeW = 1, mazeH = 1;  
static int steps = 0;  
  
static void pathFinder (int x, int y, int pathSofar)  
{  
    if (maze [y].charAt (x) == '0')  
    {  
        steps = pathSofar;  
        if (x + 1 < mazeW)  
            pathFinder (x + 1, y, pathSofar + 1);  
        if (y + 1 < mazeH)  
            pathFinder (x, y + 1, pathSofar + 1);  
    } // end if  
} //end pathFinder
```

- 1.4.3 Modify the text file, so that the path may go up and to the left as well. In order to avoid needless zigzagging by the program, modify the **pathFinder()** method so that can't move in the direction it just came from.
- 1.4.4 Add several possible other paths and branches to the original path in the text file above. Add the letter T, which stands for "treasure". It is now possible to find the shortest path to the treasure. It is now unfortunately possible to go around in circles. Therefore create a check, where you leave the recursion after a maximum number of steps have been reached.
- 1.4.5 Modify the text file, so that there is only one entrance from the outside. I.e. the first and last rows and the first and last column all contain 'x's except in one place; the entrance, Modify the program to find the entrance first.
- 1.4.6 Prepare a text file for simulating flood filling a given closed shape. Then write a program that uses recursion to fill the inside of such a shape, with 'T', given any interior point. Print the file before and after the fill. For example (next page):

```

40
15
.....TTTTTTT.....
....T.....TTTTTTTTTTTT.....
...T.....TTTT.....
..T.....T.....
..TT.....TT.....
.T.....T.....
.T.....TTT.....
..TT.....T.....
...T.....T.....
....T...TT.....T.....
.....T...TT...TT.....T.....
....T...T...T.....TTT.....
...T.....T...TTT...TTTT.....
....T.....T.....TTTTT.....
.....TTTTTT.....

```

1.4.7 Modify the flood fill method, so that it can fill with the character of your choice, and modify the program to fill the outside with one letter and the inside with another.

1.4.8 Santa's magic sack, for carrying gifts for children down the chimney, can hold up to 8 000 000 cubic centimetres. The sack can wrap around any number of gifts of any shape or size as long as it does not exceed the 8 000 000 cm³ limit.

The number of gifts, N that he can choose from does not exceed 15.

No gift is larger than 8 000 000 cm³.

The volume of each gift is expressed as an integer.

The input text file contains the number N followed by N integers, representing the gifts.

Write a program that will pick from among the N gifts the subset that will maximize the volume of the magic sack. Then print out the volume used.

(from the STA Online Computer Programming contest, DWITE, Dec 2002)

Description

Students research and review careers in computing, both locally and internationally, and create a database of companies. Students also begin a personal skills inventory and assess their employability skill level in order to set personal goals for skill improvement. Students investigate career paths leading to computer and information science careers. They research the educational requirements of a chosen career and look into the availability of programs in colleges, universities, and private institutions that will help them meet those requirements. Students also investigate apprenticeship and co-op programs.

Strand(s) & Learning Expectations:**Impact and Consequences**

- ICV.01 describe issues related to the ethical use of computers;
- ICV.03 identify post secondary educational opportunities leading to careers in information systems and computer science;
- IC1.01 explain the importance of the ethical use of computers in areas such as software piracy, privacy, and security;
- IC3.01 describe the range of career opportunities in computing and their lifelong learning requirements;
- ICV.04 explain the importance of employability skills and lifelong learning to information technology careers.

Prior Knowledge & Skills

Students should be familiar with the use of the computer, and since the ICS3M course is prerequisite to this course, that is invariably the case.

Planning Notes

- Contact the Guidance/Student Services Department to gather material regarding different career paths leading to careers in computer and information science. Invite them to make a presentation to the class regarding computer related programs in community colleges, universities, and private institutions.
- Contact the Co-op Department to make a presentation to the class regarding apprenticeship opportunities and co-op programs.

Teaching/Learning Strategies

Encourage students to research the specific topics you give them, and not to be distracted by other material that may flash before their eyes. Give them specific deadlines for coming up with material. When brainstorming, form groups of 4-5 students. Encourage the group to appoint a chair whose duty it is, to keep everyone on topic, to keep the conversation moving along and positive, and keeps everyone involved. The group should also appoint a secretary who will keep notes, and a spokesperson to report to the class as a whole.

Assessment:

Use student-teacher conferences to discuss initial skills inventories. Encourage students to track their (hopefully improving) skill level throughout the course.

Other suggestions:

A summative assessment in the form of one the following:

- a radio advertisement from a fictional company seeking candidates for a computer related job;
- a print advertisement on a possible future computer related career;
- a report regarding entry requirements to computer related programs at a chosen college, university, or private institution;
- a web page that has links to the different colleges, universities, and private institutions and their computer related programs.

Resources: **(web sites)**

<http://www.workinfonet.ca/>

Watch this space for updates on what's new, what's hot, and what's what in the employment world from Canada's Web portal to job opportunities, career, occupational, learning and labour market information.

<http://www.canlearn.ca/>

Welcome to canada's one-stop resource for the information and interactive planning tools you need to explore learning and education opportunities, research occupations, develop learning strategies, and create the financial plans to achieve your goals.

<http://ecoo.org/sigcs/compsci/>

Social Impact and Consequences, by Yee-Min Cha, Simon Fraser SS, Peel

<http://www.edu.gov.on.ca/eng/training/training.html>

Government of Ontario Training and Jobs

<http://www.hrdc-drhc.gc.ca/common/home.shtml>

Human Resources Development Canada

Exercises:

- 1.5.1 Read the skills handout (see appendix 1.5.1) and assess yourself under the various skills.
- 1.5.2 Research the literature, computer trade magazines, newspapers, co-op and/or guidance resources, and the Internet, and research various job descriptions and their lifelong learning requirements for occupations/professions in computing.
- 1.5.3 Create a personal database of company names, addresses, and phone numbers that are of particular interest to you to begin a resource list of potential contacts in the computer industry.
- 1.5.4 Compare the entry requirements to computer related programs in community colleges, universities, and private institutions. Distinguish between and compare apprenticeship and co-op programs.
- 1.5.5 Brainstorm in groups of 4-5 on the importance of having computer skills (e.g. consider the skills that your parents are required to have at their workplace)

Unit 2: Data Management Techniques

Time: 30 days

Unit Description

In this unit students will explore and expand their knowledge of Applets, They will be introduced to Classes that can be instantiated. From Objects that are visual, students will move to record classes and classes that control sets of records and binary files. Several Sorting algorithms will be developed and studied, and their relative efficiency examined.

Unit Overview Chart

Cluster	Expectations	Assessment	Focus	# days
1	TFV.02, TF1.05, TF2.01, SPV.03	K/U, A	Applets and Objects	6
2	TF1.03, TF2.02, SPV.05, SP1.06, SP2.02	K/U, A	Data Management Techniques: Binary Files	6
3	TF1.04, SP1.07, SP2.04, SP2.07, SP2.10	K/U, T/I, A	Data Management Techniques:	12
4	TFV.02, TF1.06, SP1.03	K/U, T/I, A	Sorting and Searching Data Management Techniques; Linked Lists	6

K/U = Knowledge/Understanding
T/I = Thinking/Inquiry

C = Communication
A = Application

2.1 Applets and Objects

6 days

Description

Students will be introduced to classes without the main method. They are objects that contain both data and methods to manipulate the data. In this section, object will correspond to visual shapes on the screen, such as dice objects, ball objects and face objects.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.02 explain data structures and their processing algorithms;

TF1.05 describe the difference between procedural and object oriented programming;

TF2.01 describe how procedural and object oriented programming paradigms can be used to solve different problems;

Skills and Processes

SPV.03 implement advanced data structures and algorithms;

Prior Knowledge & Skills

Students will be expected to have programmed using applets. They should be familiar with the graphics environment.

Teaching/Learning Strategies

Because of the visual nature of the objects, students will have little trouble relating to them. What may cause trouble is the need for treating the Graphics environment as a "variable" to be introduced into methods as a parameter.

There are a lot of ideas to be covered:

- the need for the object's methods to control its own data
- the idea of a constructor
- overloading (giving various options on what parameters to enter for a given method)
- inheritance (creating a new object by adding or changing properties to a given class)

If you are using Holtsoft's **Ready** software, the following programs can be written using the **hsa.Console** and need not be written using Applets.

Resources

(book)

- Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Chapter 14: **Advanced Object Oriented Concepts**
- Vincent Lo. *Learning Object Oriented Programming with Java*, 2002. ISBN 0-9695844-2-4, Unit 9: **Applets on the Web**.
- Vincent Lo. *Learning Object Oriented Programming with Java*, 2002. ISBN 0-9695844-2-4, Unit 10: **Animation**.

(web site)

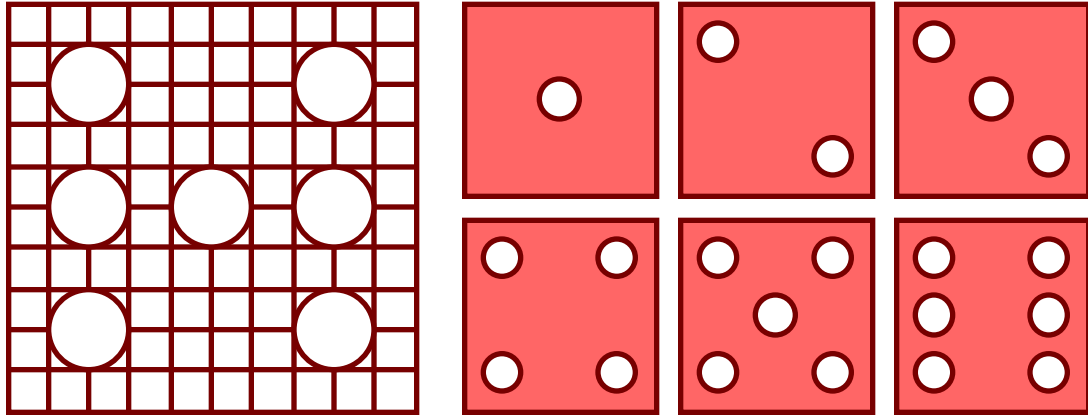
Lesson: Object Oriented programming Concepts

<http://java.sun.com/docs/books/tutorial/java/concepts/index.html>

- [Introduction to Computer Science using Java](http://chortle.ccsu.edu/cs151/cs151java.html)
chortle.ccsu.edu/cs151/cs151java.html

Exercises:

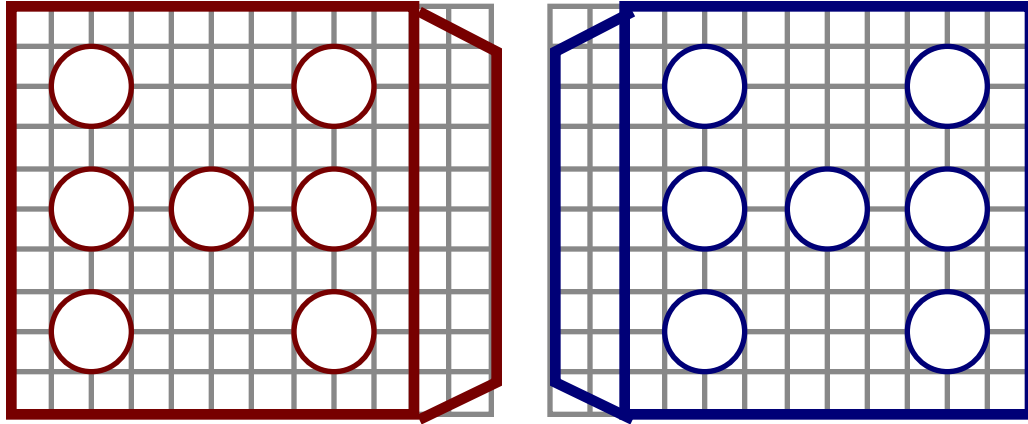
- 2.1.1 Write a program that draws a die on the screen, with the value based on a random number, using the specifications below:



in this program, the grid is composed of squares of dimensions 10x10, for a total of 100x100 pixels per die. The upper left corner be at (100,100). Write the method void dot (Graphics gr, int x, int y) where dot (g, 1,1) would correctly draw the upper left circle, and dot(g,4,4) the centre circle.

- 2.1.2 Generalize the above problem by writing a method void die(Graphics gr, int value, int x, int y, int size, int colour). It would draw a die with upper left corner at (x,y) and where **size** represents the size in pixels of the length of one square of the grid.
- 2.1.3 Create the class Die with constructor Die()
Create instance variables for the upper left corner of the die, its size, value and colour, with suitable default values.
Create the methods void setColour(Color c), void setSize(int s), void setValue(int v), setPosition(int x, int, y) and draw(Graphics gr);
Write a program to thoroughly test the class.
- 2.1.4 Write an applet, which lets you roll two dice, repeatedly until you roll a double, and you win, or a sum of 7 and you lose. Use the Die class and one button to cast the two dice.
- 2.1.5 If the dice never need to move, it is more convenient to have constructor take care of the x,y and size values. Add therefore an extra constructor to the Dice class, to accept three parameters: Dice(int x, int y, int size) . Modify 2.1.4 to test whether both constructors work.

- 2.1.6 Create an **extension** of the Die class called ThreeDeeDie.
(If the variables of the Die class were classified as "private", change them to "protected")



Add a parallelogram as shown to either the left or the right in the draw method and two corresponding methods: drawLeft() and drawRight()
Then modify program 2.1.4 to test the class.

- 2.1.7 Create the **ClickBall** class, that has the **size** property: 1=4 pixels, 2=8 pixels, etc. and that can move in any direction . The centre of the ball is **x,y** . Each time you click the screen, the ball moves 10 pixels in the direction of the click
Consider what methods **ClickBall** must have and test it with an applet.
- 2.1.8 Create the **Ball** class with constructor Ball(int maxX, int maxY), where the two variables denote the size of the applet, and with methods void move(Graphics g), and various other methods to set the size, colour, position direction and speed of the ball. Each time the move method is invoked, the position variables (x,y) should be updated and the ball painted in its new position. Then write and run the following Applet:

```
// C12p2108 class
// make sure the width of the applet is 300x300
import java.applet.*;
import java.awt.*;
public class C12p2108 extends Applet implements Runnable
{
    Thread run1;
    Ball ball1 = new Ball (300, 300);
    Ball ball2 = new Ball (300, 300);

    public void init ()
    {
        ball1.setColor (Color.blue);
        // distance 7 pixels further, 20 deg standard angle
        ball1.setSpeed (7, 20);
        ball1.setLocation (10, 100);
        ball1.setSize (2);
    }
}
```

```

        ball2.setColor (Color.red);
        ball2.setSpeed (5, 110);
        ball2.setLocation (100, 200);
        ball2.setSize (3);
    } //end init

    public void start ()
    {
        if (run1 == null)
        {
            run1 = new Thread (this);
            run1.start ();
        }
    } //end start

    public void stop ()
    {
        if (run1 != null)
        {
            run1.stop ();
            run1 = null;
        }
    } //end stop

    public void run ()
    {
        while (true)
        {
            repaint ();
            try
            {
                Thread.sleep (10);
            }
            catch (InterruptedException e)
            {
            }
        }
    } //end run

    public void paint (Graphics g)
    {
        ball1.move (g);
        ball2.move (g);
    } // end paint
} //end C12p2108

```

2.1.9 Extend the **Ball** class to create the Face class, where you add two eyes and a mouth to the ball. Modify the program 2.1.8 to bounce around some Face objects.

2.2 Record and Binary Files

6 days

Description

Students will be introduced to the concept of a record, which is a user defined data structure and, in Java, separate objects. They will expand their knowledge of data files by manipulating binary files.

Strand(s) & Learning Expectations

Theory and Foundation

- TF1.03 identify similarities and differences among data structures, including arrays, records, and arrays of records, and their applicability to solving programming problems;
- TF2.02 describe how user-defined types and records provide more flexible and powerful ways of handling data;

Skills and Processes

- SPV.05 use file management techniques in project settings.
- SP1.06 design algorithms to incorporate data structures in projects;
- SP2.02 employ user-defined data types and record data types to improve program efficiency;

Prior Knowledge & Skills

Students must have some familiarity with Objects
Students must be familiar with text files

Teaching/Learning Strategies

In Non object oriented languages Records are user defined structures that are defined as part of the main program. Record fields can be accessed directly. In Java also, fields can be accessed directly, provided the data is not declared **private**. Such a record has no methods, only an empty constructor. It may be useful to start this section by defining a record this way. (see 2.1.1)

Java does not use pointers explicitly, however, whenever memory space is allocated by invoking the **new** statement, the record named points to the new space. If a and b are records, then the statement a=b will simply cause "a" to point to the same space as "b". It is therefore crucial to assign "b" a new space to point to, before changing the values of its fields, for without the **new** statement, the field values of "a" will also appear to have changed accordingly.

Emphasize the difference between text files and binary files. Binary files have their disadvantages (list them), but their one main advantage is that the entire file need not be loaded into memory, only individual records. One may even extract a list of selected fields, without disturbing the rest of the file.

This is also an opportunity to revisit comparing simple data structures and their sizes: a byte, a character, an integer, etc. and how to convert from one to the other.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Chapter 15: **Records in Java**

(web)

Using Random Access Files
<http://java.sun.com/docs/books/tutorial/essential/io/rafIO.html>

Exercises:

Presentation - Computer Science ICS4M - John Ketelaars

p.27

Exercises:

2.2.1 Revisit the text file of 1.3.4. and create a special Record class, called **MyRecord.java**:

```
// The "MyRecord" class.
public class MyRecord
{
    String code;
    String province;
    double year [] = new double [5];

    //constructor
    public MyRecord ()
    {
    } //end constructor
} // MyRecord class
```

Then insert an extra column containing the official 2-letter provincial codes (use NU for Nunavut) and read the file with the following program (make sure both programs are saved in the same directory)

```
// The "C12p2101" class.
import java.awt.*;
import java.io.*;
import hsa.TextConsole;

public class C12p2101
{
    static TextConsole cs = new TextConsole ();
    static MyRecord list [] = new MyRecord [12]; //create a new array

    public static void main (String [] args) throws IOException
    {
        readText ("population.txt");
        cs.print ("enter a provincial code: ");
        String c = cs.readLine ();
        int location = -1;
        for (int i = 0 ; i < 13 ; i++)
            if (c.equalsIgnoreCase (list [i].code))
                location = i;
        if (location > -1)
        {
            cs.println("Population expressed in thousands");
            cs.println ("Province          "+"
                "1997      1998      1999      2000      2001");
            cs.print (list [location].province, 25);
            for (int i = 0 ; i < 5 ; i++)
                cs.print (list [location].year [i], 10, 1);
            cs.println ();
        }
    } // main method
}
```

```

public static void readText (String fileName) throws IOException
{
    String line;
    FileReader f = new FileReader (fileName);
    BufferedReader input = new BufferedReader (f);
    MyRecord prov;
    for (int i = 0 ; i < 13 ; i++)
    {
        prov = new MyRecord (); // create space for a new record
        line = input.readLine ();
        prov.code = line.substring (0, 2);
        prov.province = line.substring (3, 30).trim ();
        prov.year [0] = Double.valueOf (line.substring (30, 40)).doubleValue ();
        prov.year [1] = Double.valueOf (line.substring (40, 50)).doubleValue ();
        prov.year [2] = Double.valueOf (line.substring (50, 60)).doubleValue ();
        prov.year [3] = Double.valueOf (line.substring (60, 70)).doubleValue ();
        prov.year [4] = Double.valueOf (line.substring (70)).doubleValue ();
        list [i] = prov;
    } // end i-loop
    input.close ();
} // end readText
} // C12p2101 class

```

- 2.2.2 It is bad OOP practice to let another program directly access one's data. It is standard practice to have methods deal with this:
 Create a new Record class called **PopRecord**, and add methods that will pass the information to and from the record, e.g.:

```

public void putCode(String c)
{
    code = c;
}
public String getCode(String c)
{
    return code;
}

```

Then modify problem 2.2.1 appropriately:

Replace for example `prov.code= line.substring(0,2);`
 by: `prov.putCode(line.substring(0,2));`

- 2.2.3 Convert the filereading method to a separate class also with constructor `ReadProvPops(String filename)` and one extra method: `PopRecord getOne (int i)` that will pass on to the main program the i-th record.
- 2.2.4 Write a program that will convert the text file above into a binary file. Allocate 2 bytes to the code, 28 bytes to the province's name, and 8 each for the double year values for a total of 70 bytes each record. Use the `length()` method to find the total length of the file.

- 2.2.5 Create a program that will let you read from from the binary file and print out any record, given the code. For this, create two arrays at the start of the program, one to hold a list of codes and one to hold the corresponding (long) pointers to the start of the record that holds the associated record. (Do not load the records into an array, but read them from the file as needed). Write methods that will (1) create the data for the arrays and (2) read the appropriate record and converts it to a **PopRecord**.
- 2.2.6 Copy and paste the data from the Population of metropolitan areas into a text file, adding a column of provinces, where these cities reside, getting rid of the provinces in brackets. Then convert it to a binary file, by modifying one of the previous programs.
(<http://www.statcan.ca/english/Pgdb/demo05.htm>) Then write a program that lets you enter one letter and that prints out the data for every city that starts with that letter. Let your program stop when "X" is entered. Input should be insensitive as to case.
- 2.2.7 Create a file class general enough to be able to read both binary files: the one you created in 2.2.5. and the one in 2.2.6
the constructor: `BinaryPopFile(String fileName)` which opens the file and calculates the number of records based on the length of the file.
`int getRecNumber()`
`PopRecord getRecord(int r)`
and `close()`
Then rewrite both programs to use `PopRecord`.
- 2.2.8 To `PopRecord` add a method that lets you write a new record to the bottom of the metro population file. Then add the populations of:
Iqaluit (Nunavut), population: 4.4, 4.6, 4.8, 5.0, 5.2
Yellowknife (NT) population: 17.2, 17.1, 16.9, 16.7, 16.5
Charlottetown (PE) population: 32.5, 32.4, 32.3, 32.2, 32.2
- 2.2.9 Add a method, that lets you "delete" a record, by replacing the code of the record by "XX". modify the `getRecord()` method to ignore all records with a code of "XX". Write a program that lets you manage the metro data file, by giving you an option to add, delete, or view records.
- 2.2.10 Write a program that will "clean up" the metro binary file, by loading all valid records into an array, and resaving the data into a new file of the same name.

2.3 Sorting and Searching Techniques

12 days

Description

Students have sorted sets of data before. In this section they will be looking at several new sorting techniques, and they will examine the relative advantages and disadvantages of each. In this section the emphasis is on the non-linear sorts such as Shell sort, Merge Sort and Quick sort. Students will revisit the binary search routines, but applied to binary files and records.

Strand(s) & Learning Expectations

Theory and Foundation

TF1.04 evaluate the efficiency of different algorithms and their applicability to solving the same programming problem;

Skills and Processes

SP1.07 ensure program correctness by developing a complete suite of test data (valid and invalid data) to eliminate syntax, runtime, and logic errors;

SP2.04 build and maintain a small software library to facilitate the reuse of code;

SP2.07 compare the effectiveness of several algorithms for solving the same problem;

SP2.10 perform line-by-line walk-throughs of computer programs that include all program structures;

Prior Knowledge & Skills

Students should be familiar with binary files and tables
They are expected to be able to create appropriate records

Planning Notes

There are some good animation programs about, that will help visualize various sorting techniques. See below under Resources.

Teaching/Learning Strategies

Introduce students to each new sorting technique by letting them sort the same sample set(s). For a "walkthrough" it is useful to create a paper exercise, (see appendix 2.3.1)
Use the current time (see the System class) to compare each Sort's efficiency
Develop formulas which will explain why some sorts are more efficient than others.

Assessment & Evaluation of Student Achievement

test students knowledge about the strengths and weaknesses of each Sort
give them a paper and pencil exercise to walk through each Sort's algorithm
give students a set of problems to solve on their computer.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed.
Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Chapter 16: **Algorithms for Sorting Lists**

(Web Site)

- Dr. R. Mukundan's Applets Centre:
<http://www.cosc.canterbury.ac.nz/people/mukundan/JavaP.html>
- Sorting Algorithms
<http://www.cs.brockport.edu/cs/javasort.html>

Exercise Programs:

- 2.3.1 Write a program that lets you create a binary file that contains random integers between 0 and 9999. The name and number of items to be saved in the file is to be determined by keyboard input.
- 2.3.2 Write a program that loads the file of 2.3.1 into an array and sorts them using Bubble sort. Print the sorted array, 15 numbers per line.
- 2.3.3 Write a program that loads the file of 2.3.1 into an array and sorts them using Selection sort.
- 2.3.4 Write a program that loads the file of 2.3.1 into an array and sorts them using Insertion sort.
- 2.3.5 Write a program that loads the file of 2.3.1 into an array and sorts them using Shell sort.
- 2.3.6 Write a program that loads the file of 2.3.1 into an array and sorts them using Heap sort.
- 2.3.7 Write a program that loads the file of 2.3.1 into an array and sorts them using Quick sort.
- 2.3.8 Write a program that converts each of the above Sorts into classes that can be instantiated. Then write a program that lets you choose under input control, which of the sorts you wish to use.
- 2.3.9 Modify 2.3.8 to use **long System.currentTimeMillis()** in measuring the time each sort method takes.
- 2.3.10 Write a program that will read, sort and save the binary data file of 2.3.1
- 2.3.11 Write a class that can read and write specific integers from the data file of 2.3.10. The file should contain the constructor that will open the file for reading and writing, with input parameter the name of the file; It should contain a method for closing the file; for reading from the i-th data item and writing to the i-th data item. Write a program that will instantiate the class and test it.
- 2.3.12 Write a program that will search the file of 2.3.10 for specific numbers entered by keyboard, using a linear search. (Use the class you created in 2.3.11)
- 2.3.13 Write a program that will search the file of 2.3.10 for specific numbers entered by keyboard, using a binary search. (Use the class you created in 2.3.11)

2.4 Linked Lists

6 days

Description

Students will continue their examination of complex data structures by studying Linked Lists and thereby gain an understanding of the function of records as pointers. This section may be expanded by adding the topics of doubly linked lists and binary trees.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.02 explain data structures and their processing algorithms;

TF1.06 explain the levels of program correctness: syntax errors, runtime errors, valid data, invalid data, robustness.

Skills and Processes

SP1.03 incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;

Prior Knowledge & Skills

Students should be familiar with records, objects, binary files and recursion.

They should have a good insight into pointers, even though in Java they are not explicit.

They must have developed a good skill to analyze their runtime errors.

Teaching/Learning Strategies

Go slow: This topic tests students' analytical abilities

Explain this topic in terms of pointers

In the literature, nodes are not treated as traditional objects, but as records with public fields and no methods: The fields are directly accessible by the main program (as in C++). It makes their manipulation much simpler. In the solution set however, the pure "OOP" approach is used.

Assessment & Evaluation of Student Achievement

Test student understanding with many short conceptual questions.

In a project, give them lots of time to check runtime errors.

Accommodations

All the ministry expectations for this section are covered in other sections

It is therefore possible to skip the topic entirely or to suitably simplify it

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed.

Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4

Chapter 17: **Self-Referential Classes and linked Lists**

(web site)

- Powerpoint presentation on singly linked and doubly linked lists
[http://www.cis.upenn.edu/~matuszek/cit594/ Slides/linked-lists.ppt](http://www.cis.upenn.edu/~matuszek/cit594/Slides/linked-lists.ppt)

Linked Lists

http://www.ib-computing.com/linked_lists.htm

Exercises:

- 2.4.1 Create a linked list record class (**LinkedList.java**) which contains one string data field (name) and one pointer (LinkedList) . It should contain an empty constructor as well as the two get and two put methods for the data. Then create a test program that lets you enter several names and print them out from first to last.
- 2.4.2 Prepare a new binary file from the metro population file of problem 2.2.6, that contains only the city's name and the current (2002) population. Save the data in alphabetic order. (We will use this file in the next few programs)
- 2.4.3 Create a linked list record class called **PopMetro.java** which contains one name and one real number. Then write a program that uses it to load a linked list from the binary file of 2.4.2 and that prints the linked list on screen.
- 2.4.4 Modify the program so that it lets you add a record to the bottom of the linked list. Print the entire list, to verify the name has been added. Use for example:
Iqaluit (Nunavut), population: 5.2 Yellowknife (NT) population: 16.5
Charlottetown (PE) population: 32.2
- 2.4.5 Modify 2.4.3 so that the additions are added in alphabetical order. In order to add records to a linked list, you must consider several special cases. What happens if your addition is to be placed first? or last? What happens if the list contains no records yet?
- 2.4.6 Modify the program so that you may delete a record from the list. Search for the record by (city) name. If the city does not exist, print the appropriate message. Print the entire list, to verify the name has been appropriately deleted. Again there are special cases to be considered.
- 2.4.7 Create a class ListMetroPop that uses the routines you have developed in the previous few programs, to manage the binary file. It should contain a constructor, **ListMetroPop(String binFile)** that loads the binary file into a linked list and that contains the following methods:
`PopMetro find(String name) // null, if not found`
`void add(String name, double pop) // in alphabetic order`
`void delete (String name) // does nothing if not found`
`void print (Console c) // print the entire list on the given console`
`void save(String binFile) // saves the data to the said file`
Write a program to test the class.
- 2.4.8 Prepare a binary file based on the population on selected countries:
<http://www.statcan.ca/english/Pgdb/People/Population/demo01.htm>
One name and one number, and test the program

Unit 3: Managing Software Projects

Time: 16 days

Unit Description

Students examine the components of a software project plan and follow a carefully thought out example of a plan, in the context of case studies. They review the components of the software design life cycle and explore project management and team-building techniques. They also identify the skills that individuals contribute to the skill set of the team. Students create a list of questions, pose the questions to a role-playing client, and write a problem definition and analysis report based upon the answers.

Unit Overview Chart

Cluster	Expectations	Assessment	Focus	# days
1	TFV.01, TF1.01, IC3.04	K/U	Managing a Project: Software Design Life Cycle: Who is on the team?	3
2	SP1.01, SP2.01, IC2.03	T/I, C, A	Problem Definition and Analysis -What's the problem?	3
3	SP1.02, SP1.03	T/I, A	Design -What's the plan? Using skills of team members	4
4	SP1.03, SP2.01, IC3.02, IC3.03	C, A	Implementation – Here's the answer	4
5	SP1.07, SP2.01	A	Testing/Maintenance - Is it the right answer? Are there	2

K/U = Knowledge/Understanding
T/I = Thinking/Inquiry

C = Communication
A = Application

3.1 The Software design life cycle.

3 days

Description

Students examine the components of a software project plan and develop a plan, in the context of case studies, without coding a solution. They review the components of the software design life cycle and explore project management and team-building techniques. They also identify the skills that individuals contribute to the skill set of the team and create a list of questions, pose the questions to a role-playing client, and write a problem definition and analysis report based upon the answers.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.01 describe the steps in the software life cycle (problem definition, analysis, design, implementation, testing, and maintenance);

TF1.01 describe the components of the software life cycle and their importance in project settings;

Skills and Processes

SP1.01 devise a plan for a large software project (e.g., an accounts receivable or a random walker program), outlining the required activities at each stage of the software life cycle

Impact and Consequences

IC3.04 describe the elements of working effectively in a team environment (e.g., conflict resolution, time management, constructive criticism, task assignment).

Teaching/Learning Strategies

One of the main features of this section is the development of teams that function as a unit. when brainstorming, give students the discussion topics a day before. Let them (in groups of 4 or 5) choose a chair and a secretary. Encourage them to rotate these jobs. The chair keeps the discussion on track, interjecting "seed" questions when necessary, and to make sure every one stays involved. The secretary will record all the ideas (possibly on the computer). The job of everyone else is to contribute to the discussion. Discuss with students that the function of brainstorming is to generate ideas. Avoid negative judgments, allow for free association of ideas, use the ideas of others to piggyback for new ones of your own. Go for quantity: It is always easy to trim back later.

Plan for a definite time, usually 15-20 minutes, at the end of which the secretary transcribes the notes as quickly as possible.

Debrief each group in class: Each group will therefore appoint a spokesperson as well.

In order to familiarize students with each other, it would be a good exercise to change the group composition several times in this unit.

Assessment & Evaluation of Student Achievement

As a teacher walk around and listen to the discussion that is taking place.

Resources (web sites)

ISO/IEC 12207 Software Lifecycle Processes
<http://www.stsc.hill.af.mil/crosstalk/1996/aug/isoiec.asp>

David F. Redmiles's Requirements of the Software cycle.
<http://www.ics.uci.edu/~redmiles/ics121-FQ99/lecture/six/index.htm>

Exercises:

- 3.1.1 Your teacher wants to organize and track the computer related equipment in the computer labs and computer office. Some pieces break down and have to be replaced, or get fixed, or lend out for a certain time. In small groups, briefly discuss what takes place in each of the steps of the design life cycle with a specific example in each: (1) problem definition, (2) analysis, (3) design, (4) implementation, (5) testing, and (6) maintenance;
- 3.1.2 Your software development team thinks it may be a good idea to create a database of web sites on a given course, and develop a procedure for keeping it up to date. In a small group, brainstorm ways in which the software that maintains the database could be made as attractive as possible. Then develop a questionnaire to assess the potential need for such software.
- 3.1.3 Your software development team thinks it may be a good idea to create a calendar that can be easily updated, changed and published on the school web site, where teachers write up homework, assignment deadlines and test dates for their courses. In small groups brainstorm ways in which the software that maintains such a calendar could be made as attractive as possible. Develop questions to ask teachers to assess the need for such software and to see what the interface between teacher and calendar must look like.
- 3.1.4 Your software development team thinks it may be a good idea to create a multiple choice database that scrambles the order of questions and the set of choices for answers, so that students are tested in such a way that the person next to you can't cheat. Analyze the problem by discussing: What must the system do; how long does it take to train teachers to use the software; what hardware is required; what software must be developed; what data is needed: who will provide it; will it be secure; what is the output: what will be the format.
- 3.1.5 Consider the problem of 3.1.1, where the assignment is to create a software solution for the organizing and tracking of computer equipment in the computer labs and the computer office. Discuss in small groups, the jobs that need to be performed in each of the stages of the design life cycle: Throughout the process the team must meet to discuss the progress of the different parts. A log must be maintained; at some point a time schedule must be made up that is to be followed and if necessary modified. Reports must be written at various stages: A proposal, a progress report (possibly several, indicating the problems that have been encountered and what has been accomplished, problems foreseen, revision of the plan etc.) In the design stage, flow charts must be created, screens designed for input and output. Various functions designed for the user(s) Test data must be created...

3.2 Problem Definition and Analysis

3 days

Description

Students will be given a problem to solve. In small discussion groups they will go through the first two stages of the software design cycle: Problem Definition and Problem Analysis. They will review a sample analysis and discuss its appropriateness.

Strand(s) & Learning Expectations

Skills and Processes

- SP1.01 devise a plan for a large software project (e.g., an accounts receivable or a random walker program), outlining the required activities at each stage of the software life cycle;
- SP2.01 use an integrated development environment to create and manage a project;

Impact and Consequences

- IC2.03 evaluate the pros and cons of moving to new hardware and software technologies (e.g., costs, training requirements, compatibility, deployment);

Planning Notes

Read sections A.1-A5 of the Library Automation System Case Study found in the Appendix (pp 560-563) of **Introduction to Programming in Java**

Teaching/Learning Strategies

One of the main features of this section is the development of teams that function as a unit. when brainstorming, give students the discussion topics a day before. Let them (in groups of 4 or 5) choose a chair, a spokesperson and a secretary. Encourage them to rotate these jobs. The chair keeps the discussion on track, interjecting "seed" questions when necessary, and to make sure every one stays involved. The secretary will record all the ideas (possibly on the computer). The job of everyone else is to contribute to the discussion. The spokesperson will report to the class.

Discuss with students that the function of brainstorming is to generate ideas. Avoid negative judgments, allow for free association of ideas, use the ideas of others to piggyback for new ones of your own. Go for quantity: It is always easy to trim back later.

Plan for a definite time, usually 15-20 minutes, at the end of which the secretary transcribes the notes as quickly as possible.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Appendix A: **Library Automation Case Study.**

- **Exercise Problems**

You are asked to design a Library Automation System for a typical high school. At this point, money is no object. You will hire a team of crackerjack programmers if needed.

- 3.2.1 Using your experience with libraries in general as a guide, discuss how the current library systems you know could be improved. Create a set of questions you might use to interview librarians in your neighbourhood and school, to discover the weaknesses, problems, inefficiencies, bottlenecks in their present system. Make an inventory of all the services a library provides.
- 3.2.2 Brainstorm in small groups the requirements of the system. At the end of the brainstorming session, come up with a list of functions of the program, and divide these functions into categories. At this point, set no limits to your program.
- 3.2.3 In small groups, identify the data that is to be input, stored and outputted, and what format(s) they have to take (databases, records, fields). At this point, set no limits.
- 3.2.4 Create a flowchart (or several) that tracks the flow of data. Discuss the routines that must be developed in order to operate these functions, and the personnel needed.
- 3.2.5 Create a report of what your team's vision of the perfect solution. Include diagrams and flowcharts.

3.3 Program Design

4 days

Description

Students will be following the problem definition and the analysis of the example in the Appendix of **Introduction to Programming in Java**. They will follow the steps in the design process and analyze and critique the decisions made by the author in order to further their understanding.

Strand(s) & Learning Expectations

Theory and Foundation

TF3.02 describe the issues involved in maintaining a software library (e.g., access, backup, version control);

Skills and Processes

- SP1.02 use industry-standard methodology (e.g., flow chart, pseudocode, structure chart) in the design process;
- SP1.03 incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;

Prior Knowledge & Skills

Students should be familiar with Object Oriented Programming structures. They should be familiar with and have discussed issues surrounding creating a Library system.

Planning Notes

Review Section A2-A.6 (pp. 562 -602) of **Introduction to Programming in Java**.

Teaching/Learning Strategies

In this section, students should follow the reasoning behind the decisions made in these pages. It is therefore important that the teacher highlight the various themes that are being followed. Students will have already analyzed the problem from a slightly different perspective in the previous section:

In this section, the decision has been made that the IO interface is to be as simple as possible: Input is to be entirely by keyboard, output will on the default screen. Modularity however will guarantee that the IO interface could readily be replaced by something more sophisticated.

The functions decided upon in the book may or may not be the same as those students had chosen in the previous exercises, however, they must remain aware of the possibility of expanding the objects' data and methods in the future.

In the book objects and classes are created in a hierarchy for reasons students may not be familiar. The design process from function to classes, keeping flexibility in mind can only be illustrated with this specific example.

be aware of the processes followed when something goes wrong: when files won't open or when input entered is not as expected.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Appendix A: **Library Automation Case Study**.

Exercise Problems

- 3.3.1 Read section A.1 and compare the list of functions against those you have developed in your group. Create a new list, which includes yours. Put the list in a word processor, and keep your items in a different colour.
- 3.3.2 Read section A.2 and make a list of the data items mentioned (lists, cards, etc.). You might have envisaged other lists (possibly a list of CD's? Internet service?) Add your data to the list.
- 3.3.3 Read section A.4 and create a flowchart diagramming how information will flow from one object (program) to another, according to your understanding so far.
- 3.3.4 Study the list of fields on page 569, copy it in a word processor and discuss what fields and methods you might add. Add these, but in a different colour.
- 3.3.5 Read A.5.2 and for each of the 13 commands in the **handleCommmands()** method, graphically illustrate the movement of data, and the method that controls this movement. Explain the function of the constructor. Name this document: "LibrarySupervisor".
- 3.3.6 Study the list of objects in the chart on page 578-580 and in each case, graphically illustrate the fields and the data movements performed by its methods. Name this document "Objects"
- 3.3.7 Read A.5.3. graphically illustrate the movement of data, and the methods that controls this movement. Explain the function of the constructors. Name this document: "**LibraryItemManager**".
- 3.3.8 Revisit the **Objects** document of 3.3.6 and compare it to the chart of 584. Update the document based on this new chart.
- 3.3.9 Read A.5.4-A5.6 and graphically illustrate the data and methods of **Book**. Then using copy and paste and modify, graphically illustrate the data and methods of **Periodical**.
- 3.3.10 Read A.5.7 and graphically illustrate the data and methods of **StudentCard**, following a similar procedure you used for **Book**. Then using copy and paste and modify, graphically illustrate the data and methods of **StaffCard**. Be guided by the chart on page 590-591.
- 3.3.11 Read page 592 **Class Independence versus Flexibility**. Assuming that, we wish to change all output lists to a text file, (to be loaded into a word processor and printed from there). Which classes and their methods would have to be changed?

3.4 Problem Implementation

4 days

Description

Students will study the implementation part of the Library Automation System Case Study. They will review the decisions made in the design section and study how these decisions are fleshed out in the program.

Strand(s) & Learning Expectations

Skills and Processes

SP1.03 incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;

SP2.01 use an integrated development environment to create and manage a project;

Impact and Consequences

IC3.03 demonstrate communication skills (e.g., the ability to provide comprehensive internal documentation and the ability to explain program design and implementation clearly) in a team setting;

Planning Notes

Review Section A2-A.6 (pp. 562 -602) of **Introduction to Programming in Java**.

Teaching/Learning Strategies

In this section, students should follow the reasoning behind the decisions made in these pages. It is therefore important that the teacher highlight the various themes that are being followed.

Students will have already analyzed the problem from a slightly different perspective in the previous section:

In this section, the decision has been made that the IO interface is to be as simple as possible: Input is to be entirely by keyboard, output will on the default screen. Modularity however will guarantee that the IO interface could readily be replaced by something more sophisticated.

The functions decided upon in the book may or may not be the same as those students had chosen in the previous exercises, however, they must remain aware of the possibility of expanding the objects' data and methods in the future.

In the book objects and classes are created in a hierarchy for reasons students may not be familiar. The design process from function to classes, keeping flexibility in mind can only be illustrated with this specific example.

be aware of the processes followed when something goes wrong: when files won't open or when input entered is not as expected.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Appendix A: **Library Automation Case Study**.

Exercises

Read section A.6.1 **LibrarySupervisor**, A.7.1 **LibraryMain** and A.7.2 **LibrarySuperVisor**.

- 3.4.1 The try-catch commands are repeated several times in the program. Discuss the possibility to write a separate input class that will handle the input and associated error messages separately. Could the **Misc** class be extended this way? See A.7.11
- 3.4.2 Discuss the advantage of using constants as shown. Would the program remain as flexible, if numbers 1-13 were used instead, and the constants replaced by a list of comments, explaining the 13 commands?
- 3.4.3 the **while(true)** statement is used three times in the **LibrarySupervisor** class. It is the start of an infinite loop. State the conditions in each case, when the loop is exited, and when the loop is repeated.
- 3.4.4 Scan **LibrarySupervisor.java** and list all the classes accessed by the program through its methods.
- 3.4.5 The system uses only one data file, "library.dat" which is read by both the **LibraryItemManager** and the **LibraryCardManager**. The library items are located in the first half of the file and the library cards in the second part. Scan the **LibraryItemManager** constructor (in A.7.3) and the **LibraryCardManager** constructor (in A.7.7) to determine the composition of each file. Discuss the merits of having one single file, as opposed to two, or even 4 separate files.

Read section A.6.2 **LibraryItemManager** and the corresponding java file at A.7.3: **LibraryItemManager**.

- 3.4.6 List the classes accessed by the program.
- 3.4.7 List the methods where output to the screen may occur. Then update the word processor file you created in 3.3.7 and update it with this information.
- 3.4.8 Library items are stored in an array. Discuss how this affects the addition deletion and searching of an item. Is it important to keep the items in order? Is searching done using binary or linear search? Should printing out a list of items be done sorted by item ID or by author, title? Should overdue books be sorted by library card ID? What is actually done?

Read section A.6.3 and A.6.4 **LibraryItem, Book and Periodical** and the java files at A.7.4 - A.7.6

- 3.4.9 List the classes accessed by the program
- 3.4.10 Detail what is printed out in the "longlisting" method for the Book class, by tracking the classes and their methods involved.
- 3.4.11 List the Calendar methods involved and explain each function. Include a specific example that illustrates the function.
- 3.4.12 Type in the program (or use the files already located in **Ready**) and create at least 20 library items and 5 library cards.

3.5 Problem Testing

2 days

Description

Students will test the Library Automation System Case Study by creating test data and keep notes on how the program is performing under various conditions

Strand(s) & Learning Expectations

Skills and Processes

- SP1.07 ensure program correctness by developing a complete suite of test data (valid and invalid data) to eliminate syntax, runtime, and logic errors;
- SP2.01 use an integrated development environment to create and manage a project;

Teaching/Learning Strategies

Encourage students to work as a team where each member has a function to perform. Walk around and offer suggestions.

Resources

(book)

Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
Appendix A: **Library Automation Case Study.**

Exercises:

- 3.5.1 Type in the program (or use the files already located in **Ready**) and create at least 20 library items and 5 library cards.
- 3.5.2 Test each of the 13 functions thoroughly and keep a log of all your actions
- 3.5.3 Create a testing procedure that a stranger (for example a librarian) can follow. Discuss how you might improve the comments or error trapping
- 3.5.4 Brainstorm and make a list of all the improvements you would like to make.

Unit 4: Applying Project Management & Software Development Skills

Time: 21 days

Unit Description

This unit is a culminating challenge with two concurrent tasks. Students research, prepare, and present a report examining the use of information technology and its impact in the community and on the common good. They work in groups to apply project management skills, learned in Unit 4, to a case study. They also plan, develop, test, and document a software solution to a given problem (e.g., an inventory control system for a small business, a record system for a volunteer organization, patient records for a veterinary clinic). Students apply complex programming techniques and utilize software libraries.

Unit Overview Chart

Cluster	Expectations	Assessment	Focus	# days
1	ICV.01, ICV.02, ICV.04, IC1.01, IC1.02, IC2.01, IC2.02	T/I	Information technology and the community	2
2	SPV.01, SP1.01	A	Defining and analyzing the problem	4
3	TFV.05, SPV.01, SPV.04, SP1.02, SP1.03, SP1.05, SP1.06	K/U, A	Making a plan and defining the roles	6
4	SPV.01, SPV.02, SPV.03, SPV.05, SP1.03, SP1.04, SP2.02, SP2.03, SP2.06, SP2.12, SP3.02, SP3.03	K/U, A	Creating and testing a solution	6
5	SPV.01 SP2.08, SP2.09	T/I, A	Documenting the solution	3

K/U = Knowledge/Understanding
T/I = Thinking/Inquiry

C = Communication
A = Application

4.1 Information Technology and the Community

2 days

Description

The purpose of this section is to develop student understanding of the ethics of computer use and the impact of information technology on both the community, and society as a whole. Students will also explore post secondary education and career paths in computer science.

The skills and knowledge to be demonstrated by students will prepare them for an increasingly technological world. A foundation in this unit will introduce students to the excitement and opportunities afforded by this dynamic field and will begin to prepare them for careers in information technology.

Strand(s) & Learning Expectations

Impact and Consequences

ICV.01 describe issues related to the ethical use of computers;

ICV.02 describe the use of information technology and its impact in the community;

IC1.01 explain the importance of the ethical use of computers in areas such as software piracy, privacy, and security;

IC1.02 describe the essential elements of a code of computing ethics and why it is important to have and follow such a code;

IC1.03 analyse current media information relating to ethical issues in computing.

IC2.01 describe how local industries, businesses, or community groups are affected by the growing use of information technology to facilitate communication;

IC2.02 describe, using presentation software, how local industries, businesses, or community groups use computers to improve efficiency and productivity to serve their clients;

Prior Knowledge & Skills

Students are aware of the Acceptable Use Policy;

- are familiar with Internet searching;
- have basic Internet/CD/Library/Resource Centre research methods;
- have basic understanding of presentation techniques.

Planning Notes

Become familiar with the topics under discussion.

It may be useful to create a list of the most useful web sites on the topics of internet ethics and impact on society.

Teaching/Learning Strategies

Insist on each group to choose a chair, secretary and spokesperson, and encourage them to rotate these jobs.

If necessary, review the tasks for each job.

Assessment & Evaluation of Student Achievement

Most discussions will culminate in an oral or written presentation.

Assess these informally or formally, and encourage peer assessment.

Resources

(Web)

<http://ecoo.org/sigcs/compsci/>

Social Impact and Consequences, by Yee-Min Cha, Simon Fraser SS, Peel DSB.

Exercises:

Ethics

- 4.1.1 In groups of 4 or 5 students, brainstorm cheating: when writing an exam; when doing an assignment; when working in a group. How can the internet be used to cheat. Is plagiarism the same as cheating? At the end of about 15 minutes of discussion, come up with one statement (or paragraph) which defines "cheating" and plagiarism to your satisfaction to present to the rest of the class.
- 4.1.2 In groups of 4 or 5 students, develop 10 rules of appropriate behaviour when using a computer. These rules should cover respect for others, copying of materials, inappropriate material or language.
- 4.1.3 Have students research available material on "ethics and the internet" and list the 10 most common (or interesting) infractions. Then let students in groups of 2 or 3 choose one as a basis for a short (5 minute) presentation.
- 4.1.4 In groups of 4 or 5 students, discuss the following. At the end of the discussion be prepared to present your conclusions to the class.
- You have just received a greeting card from a friend through e-mail and now your hard drive has crashed. Describe ways to protect yourself from viruses.
 - Browsing through a game Internet site you notice the opportunity to buy the game CD online. You are filling out the information and as soon as you insert your credit card number, the form sends automatically. You do not receive a confirmation number or e-mail. How can you protect yourself from computer fraud?
 - You have just handed in your essay on Computer Crime and it was great. When you receive it back, you have a grade of zero because the teacher received another essay exactly the same as yours. Only one other person had your password. How can you protect yourself and are there alternatives to passwords?

Impact on society

- 4.1.5 In groups of 4 or 5 students brainstorm what you believe are major changes to Canadian society (e.g., family, schools, religious institutions, institutions of work and leisure, and government) brought about by computer technology (e.g., e-mail keeps people in closer contact with greater ease). Include ideas such as e-commerce, net security, online classes, relationships on the net, a sense of community, and global shrinkage. The goal is to look at how computer technology has improved (or disintegrated) parts of society. Develop a list of the top 3 changes you consider the most influential, and after a 15 minute discussion be prepared to present your reasons for choosing these three.

- 4.1.6 As a class, list all the changes brought forward and through a voting process rank these changes in order of importance.
- 4.1.7 Students work in pairs or on their own to come up with ideas for software programs or new technology that will enhance society. Write a one-page report, discussing your views on what would make an ideal software program from a societal point of view.
- 4.1.8 In groups of 4 or 5 students brainstorm on one topic each. After 15-20 minutes present your conclusions to the class:
- If you were developing new software for home use, what would you take into consideration that would enhance family life?
 - Electromagnetic fields have become an issue that computer users cannot ignore. As a computer manufacturer, how can you help?
 - Software piracy is a growing trend. As a developer of new software, what can you do to protect yourself?
 - As a home computer user you do not want to be worried about viruses. What new viruses are out there and what is coming?
 - You are always looking for ways to save your company money. Is developing a computer with artificial intelligence accountant a good idea?
 - You are in the field of virtual reality. Describe the new offerings for home use.
 - Your job is in the computer memory field. You must decide what information should be stored as "firmware".

4.2 Defining and analyzing the problem

4 days

Description

Students organize themselves in small groups for the purpose of doing a large software project. They will select a problem to solve and study its details for the task ahead.

Strand(s) & Learning Expectations

Skills and Processes

SPV.01 incorporate the software life cycle in project settings;

SP1.01 devise a plan for a large software project (e.g., an accounts receivable or a random walker program), outlining the required activities at each stage of the software life cycle;

Prior Knowledge & Skills

Students should be familiar with the software life cycle in project settings.

Planning Notes

Prepare a list of projects to suggest to your students.

Teaching/Learning Strategies

Once groups have chosen their project, encourage them to do some preliminary research on the topic, consult knowledgeable people and consider the need for special hardware and software.

resources:

(book)

Richard J. Kitto, *Computers and Problem Solving, a Case Study Approach*, Toronto: McGrawHill-Ryerson, 1989.

Exercises

Pick a topic from one of the following or another, with your teacher's permission: Your program must contain a data base that is to be maintained as part of the problem. **Keep in mind that you have only about 20 days to do this work.**

- 4.2.1 A mom & pop variety store have a sideline, where they rent out CD's. They have an arrangement with the publishing companies, where they are allowed to copy the CD, provided they send a \$2.00 fee for each new copy made to the company. They keep the original in a secure place and only lend out copies, which they destroy as a routine after 5 viewings. Write a program that will help keep track of all the details
- 4.2.2 A history teacher wants a piece of software that keeps a list of internet addresses about various historical events and time periods. Unfortunately topics on the internet are notoriously "temporary" and may disappear or change address at any time. They need to be checked from time to time. They also need to be classified in some easy way, new discoveries added, old ones deleted. Develop a program that will help organize such a database.
- 4.2.3 You are developing a multiple choice questions database for any high school course. Questions are classified under several headings: different topics, different difficulty levels. A student must select a topic, difficulty level and number of questions and is scored immediately upon completion of the test. Your program should keep complete statistics on each question: how often anyone answered it right or wrong. questions may be added or deleted or modified or reclassified.
- 4.2.4 Consider the site: http://www.bcca.org/misc/qiblih/latlong_ca.html
In spite of the fact that distances along the surface of the earth are in the form of arches, it is possible to find the distance between any two cities on earth: Convert minutes to degrees, consider N and S + and - respectively, consider E and W + and - relatively speaking also.
You can then use the **Haversine Formula** for calculating distances:
 $d_{long} = lon2 - lon1$
 $d_{lat} = lat2 - lat1$
$$a = \sin^2 \frac{d_{lat}}{2} + \cos(at_1) \times \cos(at_2) \times \sin^2 \frac{d_{lon}}{2}$$

$$c = 2 * \arcsin(\min[1, \sqrt{a}])$$

 $d = 6367 \times c$
where d is the distance in km.
Write a program that will create and maintain a database of cities with their longitude and latitudes and that will let you print out the distance between any pair of cities in the database. It should allow you furthermore to select one city as a reference and list the distances to any set of up to 5 other cities.

Proposal:

- 4.2.5 Research the problem you have chosen: Create a list of questions about it that you need to discuss among yourselves, ask someone(s) you consider an expert. Then write up the proposal which states the problem and outline the solution.
- 4.2.6 Brainstorm and list the various functions you wish your project to perform and group them. At this point do not set limits.
- 4.2.7 Based on your list of functions, discuss the various objects you need, and the information they should hold. Discuss as well what form output and input should take. At this point also, critically look at the functions you have listed and modify them: Classify them as: Plus-Minus-Interesting before making your decision.

4.3 Making a plan and defining the roles

6 days

Description

Students will break up the project into sections, fit it in a timetable and allocate jobs for each member of the team. They will write up a formal proposal of the project.

Strand(s) & Learning Expectations

Theory and Foundation

TFV.05 describe the relationship among hardware, software, and network requirements.

Skills and Processes

- SPV.01 incorporate the software life cycle in project settings;
- SPV.04 identify on-line and off-line resource materials;
- SP1.01 devise a plan for a large software project (e.g., an accounts receivable or a random walker program), outlining the required activities at each stage of the software life cycle;
- SP1.02 use industry-standard methodology (e.g., flow chart, pseudocode, structure chart) in the design process;
- SP1.03 incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;
- SP1.05 select appropriate data structures (e.g., arrays, records, arrays of records) for use in projects;
- SP1.06 design algorithms to incorporate data structures in projects;

resources:

(book)

Richard J. Kitto, *Computers and Problem Solving, a Case Study Approach*, Toronto: McGrawHill-Ryerson, 1989.

Exercises

- 4.3.1 Do some research into the feasibility of using special hardware or software. It may be or may not be practical to use special output or input routines. Consult your teacher or other expert; consult the internet; check what is readily available.

- 4.3.2 Allocate jobs for various stages in the creation process: One person should keep a log of all events: hurdles, problems, modifications, and progress. One person is the discussion leader. One person takes charge of activities and encourages everyone to follow group decisions. Other jobs will flow from the daily meetings. Make sure everyone has a job to do.
- 4.3.3 Create a formal proposal. The proposal should identify your group, the title and function of the software, the audience. In it, itemize the functions the software should perform, possibly include a sample input or output screen. As well, include a development plan that include such major activities as System Analysis, Creating a flowchart, Code writing, Manual preparation and Field testing (which includes creating test data)
- 4.3.4 Resolve your program into data and the methods to manipulate this data. Group the data into objects, and consider what methods need to be developed in order to pass that information forward. Typically a **record** object contains fields and methods for setting and getting the information from the fields. A **file** object typically contains sets of discrete records and methods for loading and saving to or from disk, for adding, deleting, accessing them using some identifier (searching), and sorting them. Consider other methods you may find useful: Specialized input; specialized output, other special routines. Create a list of Classes, their data and methods.
- 4.3.5 Revisit your the list of function you created in your formal proposal, and create a flowchart for each, listing among others, the data and methods that may possibly be involved. Use this information to review and modify the methods for each class, considering their input parameters and the data they return.
- 4.3.6 Decide on the limits of various data (how many records, how large each data item) and discuss expandability. Consider the functions you have rejected, could they be added at a later date?

4.4 Creating and testing a solution

6 days

Description

Each team will write the program one section at a time and test each part separately and later as a unit. They will be writing special short-term programs and create data for testing purposes. They will invite people outside the group to critique the performance of the program.

Strand(s) & Learning Expectations

Skills and Processes

- SPV.01 incorporate the software life cycle in project settings;
- SPV.02 effectively use software development and diagnostic tools;
- SPV.03 implement advanced data structures and algorithms;
- SPV.05 use file management techniques in project settings.
- SP1.03 incorporate modularity, software reuse, and maintenance considerations at the design and implementation stages of the project;
- SP1.04 incorporate appropriate code from shared software libraries into software projects;
- SP2.02 employ user-defined data types and record data types to improve program efficiency;
- SP2.03 use arrays, records, and arrays of records in different project settings;
- SP2.11 use appropriate research and resource materials to independently master new programming skills;
- SP2.12 effectively critique programs written by others;
- SP2.13 log error messages and appropriate fixes.
- SP3.02 develop software libraries in project settings;
- SP3.03 use predefined modules from software libraries to improve productivity.

Teaching/Learning Strategies

Emphasize the need for a log of all activities of the group.

The log is necessary to assess all the labour that went into the project, which in real life would translate into cost.

A log is also necessary to enable the teacher to assess student work.

Students must realize that testing is a time-consuming job, and error trapping is often very frustrating.

resources:

(book)

Richard J. Kitto, *Computers and Problem Solving, a Case Study Approach*, Toronto: McGrawHill-Ryerson, 1989.

Exercises

- 4.4.1 Create each class and test each separately with a special program and with test data specifically written for the test. Add internal documentation to all your routines, and insert the name of the person(s) who wrote the class. Modification of the class may be necessary, and the documentation will refresh your memory.
- 4.4.2 Once all the classes have been debugged separately, test them as a unit. Use "normal" test data first, and later, concentrate on "crazy" data, to test for robustness.

- 4.4.3 Develop a list of testing instructions for a "strangers" to follow, and see how well they perform. You can no longer be objective about how transparent and easy to use your program is. Modify the I/O routines if necessary.

4.5 Documenting the solution

3 days

Description

Students will create internal and external documentation. The external documentation will consist of two manuals, a technical manual and a users manual. They will also write a report based on their log, that summarizes the process of the project.

Strand(s) & Learning Expectations

Skills and Processes

SPV.01 incorporate the software life cycle in project settings;
SP2.08 produce comprehensive documentation (e.g., help files, manuals) for a software project
SP2.09 perform peer reviews of internal and external documentation;

Teaching/Learning Strategies

Emphasize to students that the process from start to finish is a growth of expertise that cannot be denied. It is important to look at the good as well as the bad decisions. The so-called "bad" decisions are only "bad" with hindsight.

resources:

(book)

Richard J. Kitto, *Computers and Problem Solving, a Case Study Approach*, Toronto: McGrawHill-Ryerson, 1989.

Exercises

- 4.5.1 Create a technical manual, that records the classes, their hierarchy, if any, and their data, constructors and functions. Briefly explain their functions. Explain how data is stored on disk
- 4.5.2 Create a report that explains the changes your program has undergone from start to finish. Highlight the struggles with concepts, the victories, when things went well, and the compromises in the face of time and resources. List the jobs done by each member of the group. Consult the log you kept throughout.
- 4.5.3 Create a users manual, that explains in nontechnical terms how to operate your software.
- 4.5.4 Present the software to the class and use a presentation software to "advertise" it.

Appendix

Accommodations

Teachers should be aware of students who have an IEP (Individual Education Plan) and ensure that the necessary accommodations are in place.

The following accommodation strategies may be used:

- provide adaptive hardware devices (e.g., large screen monitors, larger fonts, special keyboards);
- provide any environmental changes that may assist in mobility and safety in the classroom;
- continue to develop and use glossaries and word lists of key terms and phrases; make use of diagrams, posters, and handouts that support visual strengths;
- conference with the student as her/his own advocate and appropriate school personnel;
- select a case study context familiar to students to ensure better understanding of the requirements (e.g., students may develop a software package for their church or community group);
- provide opportunities for enrichment throughout the course;
- provide a choice of assignment formats and allow extra time as appropriate to needs;
- provide the following as supports to procuring marks on assessments: extended timeline, wordlists, scribing, alternative presentation format, and frequent feedback/dialogue on one-to-one/team level.

Resources

The following resources are used in many activities; specific resources are included in the developed units.

- Lo, Vincent. *Learning Object Oriented Programming with Java*, Richmond Hill, BioStrategy Inc. 2002. ISBN 0-9695844-2-4,
- Hume, J.N.P. and Christine Stephenson. *Introduction to Programming in Java*, 1st ed. Toronto: Holt Software Associates Inc., 2000. ISBN 0-921598-39-4
- Kitto, Richard J. *Computers and Problem Solving, a Case Study Approach*, Toronto: McGrawHill-Ryerson, 1989. ISBN 0-07-549826-X
- Carter, John . *Problem solving in Pascal* . Don Mills: Addison Wesley 1989 ISBN 0-201-11215-9

Postsecondary Education

Canadian Universities and Colleges from Yahoo! –

http://ca.yahoo.com/Regional/Countries/Canada/Education/Higher_Education/Colleges_and_Universities

CommuniCAAT Site (annual calendar) – <http://www.ocas.on.ca>

Government of Ontario Postsecondary Site –

<http://www.edu.gov.on.ca/eng/general/postsec/postsec.html>

Ontario Universities Application Centre – <http://www.ouac.on.ca/>

OSS Considerations

The Grade 12 Computer and Information Science course can be used to fulfill the requirement for "an additional credit in science [Grade 11 or Grade 12] or technological education credit [Grades 9-12]" (OSS, 1999, p. 9). This course provides students with educational preparation for university and college.

The curriculum emphasizes theory and concrete applications. Teaching/learning strategies and accommodations are selected to meet the needs of each student. Anti-discrimination education, accommodations for exceptional students, career goals/cooperative education, and community partnerships are addressed. These inclusions support the policies in *Ontario Secondary Schools*,

Grades 9 to 12: Program and Diploma Requirements, 1999. Career exploration is available with reference to *Choices Into Action: Guidance and Career Education Program Policy for Elementary and Secondary Schools, 1999*.

Appendix 1.5.1 Skills Inventory

(by Yee-Min Lee)

Skill Description (Academic Skills)	Level				
	1	2	3	4	5
Read and understand written material					
Write effectively					
Present own ideas to others					
Understand oral information					
Comprehend basic math					
Organize data					
Possess computer skills					
Use a calculator to solve problems					
Use tools and equipment					
Read and understand technical materials					
Accept ideas of others					
Research skills					
Learn through observation					
Make decisions					
Think independently					
Create quality products					
Generate ideas					
Learn quickly					

Skill Description	Level				
	1	2	3	4	5
(Personal Management Skills)					
Attend school daily and on time					
Meet school work deadlines					
Demonstrate self-control					
Show initiative					
Time-management skills					
Aware of safety concerns					
Work well with others					
Self-esteem and confidence					
Honest					
Integrity					
Positive attitude					
Persistence					
Respect for others					
Creative ideas					
Able to prioritize					
Upgrade skill levels					
Money management					
Sense of humour					

Skill Description (Teamwork Skills)	Level				
	1	2	3	4	5
Listen to others					
Identify non-verbal clues					
Try new things					
Dedicated					
Co-operative					
Plan with others					
Work toward group's goals					
Exercise give and take					
Leadership					
Encourage participation					
Concern for others					
Responsible					
Speaks to groups					
Identify group problems					
Negotiation					
Supportive of group					
Brainstorming					
Role model					

Appendix 2.3.1 Sorting Examples

In this example, **Bubble sort** is used, to sort the sample of numbers
Each new line represents a pass through the entire set:

5	7	3	8	1	4	6	4	2	9	7
5	3	7	1	4	6	4	2	8	7	9
3	5	1	4	6	4	2	7	7	8	9
3	1	4	5	4	2	6	7	7	8	9
1	3	4	4	2	5	6	7	7	8	9
1	3	4	2	4	5	6	7	7	8	9
1	3	2	4	4	5	6	7	7	8	9
1	2	3	4	4	5	6	7	7	8	9

In this example **Insertion sort** is used to sort the sample of numbers
Each new line represents a pass that increases the size of the sorted set by one.

5	7	3	8	1	4	6	4	2	9	7
5	7	3	8	1	4	6	4	2	9	7
3	5	7	8	1	4	6	4	2	9	7
3	5	7	8	1	4	6	4	2	9	7
1	3	5	7	8	4	6	4	2	9	7
1	3	4	5	7	8	6	4	2	9	7
1	3	4	5	6	7	8	4	2	9	7
1	3	4	4	5	6	7	8	2	9	7
1	2	3	4	4	5	6	7	8	9	7
1	2	3	4	4	5	6	7	8	9	7
1	2	3	4	4	5	6	7	7	8	9

In this example **Quick sort** is used to sort the sample of numbers
 Each new line the rectangle represents the set to be partitioned
 The circle inside represents the pivot before partitioning.
 the circles outside represent the old pivots in their final positions.

(5)	7	3	8	1	4	6	4	2	9	7
(3)	1	4	4	2	(5)	7	8	6	9	7
(1)	2	(3)	4	4	(5)	7	8	6	9	7
(1)	2	(3)	(4)	4	(5)	7	8	6	9	7
(1)	2	(3)	(4)	4	(5)	(7)	8	6	9	7
(1)	2	(3)	(4)	4	(5)	6	(7)	(7)	8	9
(1)	2	(3)	(4)	4	(5)	6	(7)	7	(8)	9
(1)	2	(3)	(4)	4	(5)	6	(7)	7	(8)	9