Family Name: . . . . . **Other Names:** . . . . . .

ID Number: . . . . . .

COMP102: Test 2

Model Solutions

26 April, 2007

Instructions

- Time allowed: **90 minutes** ($1\frac{1}{2}$ hours).
- There are 90 marks in total.
- Answer **all** the questions.
- Write your answers in the boxes in this test paper and hand in all sheets.
- If you think some question is unclear, ask for clarification.
- There is some Java documentation at the end of the test paper.
- This test will contribute 20% of your final grade, if it helps your grade.
- Non-electronic translation dictionaries and calculators without a full set of alphabet keys are permitted.

Questions	Marks	
1. Basic Java	[32]	
2. *Using a DrawingCanvas*	[-]	
3. Debugging Conditionals	[22]	
4. Objects and Fields	[12]	
5. Loops with Files	[14]	
	TOTAL:	

Student ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Basic Java [32 marks]

(a) [3 marks] What will the following fragment of Java print out?

```
String word = "Hello There!";
System.out.println("word is " + word);

String word2 = word.toUpperCase();
System.out.println("word2 is " + word2);

String message = "None";

if (word.length() > 20)
    message = "too long";
else if (word.equals("hello there"))
    message = "Yes";

System.out.println("message is " + message);
```

```
word is  Hello There!
word2 is  HELLO THERE!
message is  None
```

(b) [4 marks] What will the following fragment of Java print?

```
for (int k=8; k>=4; k=k-2){
    System.out.println("num = " + k);
}
```

```
num = 8
num = 6
num = 4
```

(Question 1 continued on next page)

(Question 1 continued)

(c) [4 marks] Consider the following drawShape method with two if statements. Assume that canvas is a field containing a DrawingCanvas.

```
public void drawShape(int n){

    canvas.setForeground(Color.red);

    if ( (n <= 1) || (n == 4) )
        canvas.setForeground(Color.blue);

    if ( (n > 2) && (n <= 5) )
        canvas.setForeground(Color.green);

    canvas.fillRect(10,10,100,100);
}
```

For each of the following four calls to drawShape, state what colour rectangle will be drawn.

drawShape(1):	blue
drawShape(2):	red
drawShape(3):	green
drawShape(4):	green

(d) [4 marks] Write a fragment of Java using a **while** statement that behaves the same as the following fragment which uses a **for** statement:

```
for ( int x = max; x >= 1; x = x-1){
    System.out.println("x = " + x);
}
```

```
int x = max;
while ( x >= 1 ){
    System.out.println("x = " + x);
    x = x-1;
}
```

(Question 1 continued on next page)

(Question 1 continued)

(e) [4 marks] Write a method called emphasise which has one *String* parameter and returns a value of type *String*. emphasise should return a new string that starts and ends with an "*" and contains the value of its parameter in the center.

For example, emphasise("NO") should return the string "*NO*".

```
public String emphasise(String arg){
    return "*" + arg + "*";



}
```

(f) [7 marks] The following sqrRoots method has one integer parameter, max. Complete the method so that it prints out the square roots of each integer from 4 up to and including max, using either a **for** loop or a **while** loop. For example, calling sqrRoots(9) should print out

```
2.0
2.23606797749979
2.449489742783178
2.6457513110645907
2.8284271247461903
3.0
```

Hint: use the Math.sqrt method to compute the square roots.

```
public void sqrRoots(int max){
        for (int j=4; j<=max; j++){
            System.out.println(Math.sqrt(j));
        }
// or
        int n=4;
        while (n<=max){
            System.out.println(Math.sqrt(n));
            n++;
        }
}
```

(Question 1 continued on next page)

(Question 1 continued)

Consider the following Compute class.

```
public class Compute {

    public void print( int n) {
        System.out.println("  print:" + n);
        int k = this.compute(n);
        System.out.println("  -> " + k);
    }

    public int compute(int n) {
        System.out.println("    compute:" + n);
        return (2 * n + 1);
    }

    public void printAll ( int n) {
        System.out.println("printAll:" + n);
        for ( int i=1; i<n; i++) {
            this. print ( i );
        }
        System.out.println("done");
    }
}
```

(g) [2 marks] What will be printed out if print(6) is called on a Compute object?

```
  print:6
    compute:6
  -> 13
```

(h) [4 marks] What will be printed out if printAll(3) is called on a Compute object?

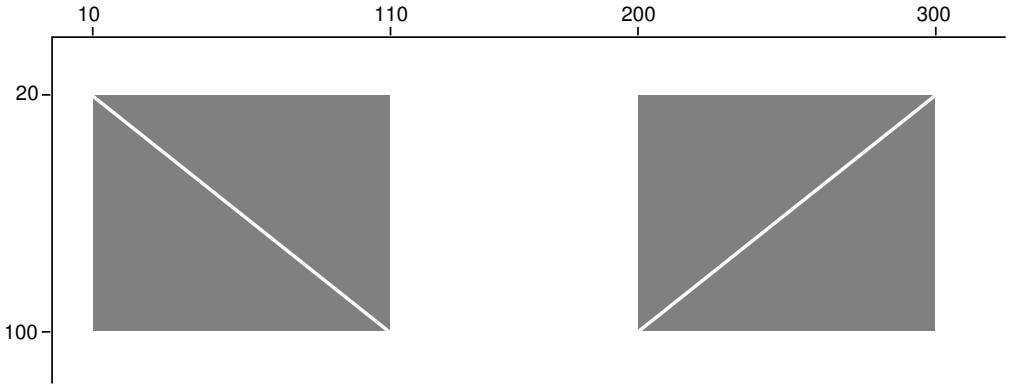
```
printAll:3
  print:1
    compute:1
  -> 3
  print:2
    compute:2
  -> 5
done
```

Question 2. Using the DrawingCanvas [0 marks]

Complete the following drawFlag method so that it draws a gray rectangle with a white diagonal line joining two opposite corners. The drawFlag method has five parameters that are the coordinates of the positions of two opposite corners: (x1, y1) and (x2, y2), and the canvas to draw on. The white line should go from one of these corners to the other.

The line will be a diagonal down from the left or a diagonal down from the right, depending on the coordinates. You should assume that x1 is always less than x2.

For example, drawFlag(10, 20, 110, 100, canvas) would draw the flag shown on the left; drawFlag(200, 100, 300, 20, canvas) would draw the flag shown on the right.



There is documentation on the DrawingCanvas and Color classes at the end of the test paper.

```
public void drawFlag(int x1, int y1, int x2, int y2, DrawingCanvas canvas){

    canvas.setForeground(Color.gray);
    if ( y1 < y2 )
        canvas.fillRect (x1, y1, (x2−x1), (y2−y1));
    else
        canvas.fillRect (x1, y2, (x2−x1), (y1−y2));
    canvas.setForeground(Color.white);
    canvas.drawLine(x1, y1, x2, y2);
  }









}
```

Question 3. Debugging Conditionals [22 marks]

The smallest method below is intended to return the smallest of four integers.
For example, smallest(8, 17, 2, 9) should return 2.

```
/* This version of smallest has errors */
public int smallest( int a, int b, int x, int y){
    if ( (a < b) && (x < y) ){
        if (a < x)
            return a;
        else
            return x;
    }
    else if ( (b < a) && (y < x) ){
        if (b < y)
            return b;
        else
            return y;
    }
    else {
        if ( a < y )
            return a;
        else
            return y;
    }
}
```

The smallest method above has errors.

(a) [3 marks] What will smallest(8, 17, 2, 10) actually return?

```
2
```

(b) [3 marks] What will smallest(17, 8, 2, 10) return?

```
10
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [6 marks] Show two further calls to smallest using the arguments 2, 8, 10, 17 in different orders, one of which will return the correct answer and the other will return an incorrect answer. Show the arguments and the answer.

```
Correct:    any of (2, 8, 10, 17)   (2, 10, 8, 17)   (2, 17, 8, 10)
  (8, 10, 2, 17)   (8, 17, 2, 10)   (10, 17, 2, 8)   (8, 2, 17, 10)
  (10, 2, 17, 8)   (17, 2, 10, 8)   (10, 8, 17, 2)   (17, 8, 10, 2)

Incorrect:  any of (17, 2, 8, 10):10   (10, 2, 8, 17):10
  (8, 2, 10, 17):8   (10, 8, 2, 17):10   (17, 10, 2, 8):8
```

(d) [10 marks] Write a correct version of the smallest method which always returns the smallest argument.

```
public int smallest(int a, int b, int x, int y){


//There are many possible versions.
        int first = a;
        int second = x;
        if ( b < a )
            first = b;
        if ( y < x )
            second = y;
        if ( first < second )
            return first ;
        else
            return second;









}
```

Question 4. Objects and Fields [12 marks]

Consider the following Student class which defines objects containing information about students, and a checkCode method for testing the code.

```
public class Student {
    private String name;
    private int labs;       // number of labs student has completed.

    public Student (String n, int m) {
        this.name = n;
        this.labs = m;
    }
    public String getName() {
        return this.name;
    }
    public int getLabs() {
        return this.labs;
    }
    public void setName(String n) {
        this.name = n;
    }
    public void setLabs(int n) {
        this.labs = n;
    }
    public void print () {
        System.out.println("Name: " + this.name);
        System.out.println("Labs: " + this.labs);
    }
    public String getPassMessage() {    // has student passed the labs requirement?
        if ( this.labs >= 7 ) {
            return (this.name + " has finished");
        } else
            return (this.name + " needs more labs");
    }
    public static void checkCode() {
        Student s1 = new Student ("Alan", 5);
        Student s2 = new Student ("Bob", 8);
        Student s3 = new Student ("Clare", 4);
        s1.print ();
        System.out.println(s2.getName());
        System.out.println(s3.getLabs());
        s1.setLabs(9);
        System.out.println(s1.getPassMessage());
        System.out.println(s3.getPassMessage());
    }
}
```

(Question 4 continued on next page)

(Question 4 continued)

(a) [6 marks] What will the following program print out when the main method in StudentReporting is called?

```
public class StudentReporting{

    public static void main(String[] args) {
        Student s1 = new Student ("Alan", 5);
        Student s2 = new Student ("Bob", 8);
        Student s3 = new Student ("Clare", 4);
        s1.print ();
        System.out.println(s2.getName());
        System.out.println(s3.getLabs());
        s1.setLabs(9);
        System.out.println(s1.getPassMessage());
        System.out.println(s3.getPassMessage());
    }
}
```

```
Name:  Alan
Labs:  5
Bob
4
Alan has finished
Clare needs more labs
```

(Question 4 continued on next page)

(Question 4 continued)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 4 continued)

(b) [6 marks] Define an addLabs method for the Student class to update the number of labs. The addLabs method should have one parameter that is the number of new labs that the student has completed, and the value of the labs field should be increased by this number. The method should also print the name of the student, the number of labs the student had previously completed, and then print the message saying whether the student has now passed the labs requirement.

```
public void addLabs(int n){
    this.print();
    this.labs = this.labs + n;
    System.out.println(this.getPassMessage());
}
```

Question 5. Loops with files [14 marks]

Suppose the file fiveHouses.txt contains the following information about five houses for sale. Each house is described on one line: each line contains the location of the house, the number of bedrooms, the number of bathrooms, the number of car parks, and the price.

```
Kelburn 4 2 1 800000
Karori 3 3 2 600000
Brooklyn 3 2 2 630000
MtCook 2 1 0 500000
Kelburn 3 2 1 600000
```

The following printHouseData method processes data from files with this format:

```
public void printHouseData(String fileName){
  try{
    Scanner fileScan = new Scanner(new File(fileName));

    String d1 = fileScan.next();
    int b1 = fileScan.nextInt();
    String d2 = fileScan.nextLine();
    String d3 = fileScan.nextLine();

    System.out.println(d3);
    System.out.println(d2);
    System.out.println(d1 + b1);
    System.out.println(fileScan.next());

    fileScan.close();
  }
  catch(IOException e){System.out.println("File reading failed");}
}
```

(a) [4 marks] What will be printed by the call printHouseData("fiveHouses.txt")?

```
Karori 3 3 2 600000
2 1 800000
Kelburn 4
Brooklyn
```

(b) [10 marks] Complete the following selectHouses method which has one argument which is the name of a text file containing data about houses in the format given above. selectHouses should read the data in the file to find all the houses with 3 bedrooms and 2 bathrooms. It should write all the other information about each such house (location, number of car parks, and price) to a file called "`bigHouses`". At the end, it should also print out (to System.out) the percentage of houses in the data file that met the criteria and the average price of those houses. Note that your method should work for any file that has data in the correct format. You may assume that there is at least one house in the file.

There is documentation on relevant classes, including Scanner, PrintStream, and File, at the end of the test paper.

```
public void selectHouses(String fileName){

    try{
        Scanner sc = new Scanner(new File(fileName));
        PrintStream fileOut  = new PrintStream(new File("bigHouses.txt"));
         int  count = 0;
         int  meet = 0;
        double totalPrice  = 0;
        while (sc.hasNext()){
        count++;
            String  location  = sc.next ();

            int  bedrooms = sc.nextInt();
            int  bathrooms = sc.nextInt();
            int  parking =  sc. nextInt ();
            int  price  =  sc. nextInt ();

            if  ( bedrooms == 3 && bathrooms == 2 ){
                meet++;
                 totalPrice  += price;
                 fileOut . printf ("%s %d %d\n", location, parking, price);
            }
            }
        sc.close ();
         fileOut .close ();
        System.out.printf("%.1f%% of houses had 3 bedrooms and 2 bathrooms\n",
                          meet*100.0/meet);
        System.out.printf("Average price = $%.2f\n", totalPrice/count);
    }
    catch(IOException e){System.out.println("File reading or writing failed");}


}
```

Student ID: .....

****

Student ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(You may detach this page)

Brief and partial documentation of some classes and methods

```
PrintStream class:                                  // Note, System.out is a PrintStream object
 public PrintStream (File f);                       // Constructor, for printing to a file
 public void close();                               // Close the file (if it is wrapping a File object)
 public void print(String s);                       // Prints s with no newline
 public void print(int i);                          // Prints i with no newline
 public void print(double d);                       // Prints d with no newline
 public void println();                             // Prints a newline
 public void println(String s);                     // Prints s followed by newline
 public void println(int i);                        // Prints i followed by newline
 public void println(double d);                     // Prints d followed by newline
 public void printf(String format, ...);            // Prints the format string, inserting the remaining
                                                    // arguments at the %'s in the format string:
                                                    //   %s for Strings.
                                                    //   %d for ints, (%3d: use at least 3 characters),
                                                    //   %.2f for 2dp doubles,
                                                    //   (%6.2f: at least 6 characters and 2dp),
                                                    // Use \n for newline

Scanner class:
   public Scanner(InputStream i);                   // Constructor. Note: System.in is an InputStream
   public Scanner(File f);                          // Constructor, for reading from a file
   public Scanner(String s);                        // Constructor, for reading from a string
   public boolean hasNext();                        // Returns true if there is more to read
   public boolean hasNextInt();                     // Returns true if the next token is an integer
   public boolean hasNextDouble();                  // Returns true if the next token is a number
   public String next();                            // Returns the next token (chars up to a space/line)
   public String nextLine();                        // Returns the next line
   public int nextInt();                            // Returns the integer value of the next token
                                                    //   (throws exception if next token is not an integer)
   public double nextDouble();                      // Returns the double value of the next token
                                                    //   (throws exception if next token is not a number)
   public void close();                             // Closes the file (if it is wrapping a File object)

File class:
   public File (String fname);                       // Constructor. Creates a File object attached to the
                                                     //   file with the name fname
Integer class:
   public static final int MAX_VALUE;               // The largest possible int: 2^(31−1)
   public static final int MIN_VALUE;               // The smallest possible int: −2^(31)

Double class:
   public static final double MAX_VALUE;            // The largest possible double: just under 2^(1024)
   public static final double MIN_VALUE;            // The smallest possible positive nonzero double
   public static final double POSITIVE_INFINITY;    // positive infinity (greater than any number)
   public static final double NEGATIVE_INFINITY;    // negative infinity (less than any number)
   public static final double NaN;                  // The Double that is Not a Number (''undefined'')
```

(Continued on next page)

String **class**:

```
public int length();                          // Returns the length (number of characters) of the string
public boolean equals(String s);              // String has same characters as s
public boolean equalsIgnoreCase(String s);    // String has same characters as s, ignoring their case
public String toUpperCase(String s);          // Returns upper case copy of string
public String toLowerCase(String s);          // Returns lower case copy of string
public boolean startsWith(String s);          // First part of string matches s
public boolean contains(String s);            // s matches some part of the string
public String substring(int j, int k)         // Returns substring from index j to index k−1
public int indexOf(String s);                 // returns the index of where s first matches
                                              // Returns −1 if string does not contain s anywhere
```

Math **class**:

```
public static double sqrt(double x);          // Returns the square root of x
public static double min(double x, double y); // Returns the smaller of x and y
public static double max(double x, double y); // Returns the larger of x and y
public static double abs(double x);           // Returns the absolute value of x
public static int min(int x, int y);          // Returns the smaller of x and y
public static int max(int x, int y);          // Returns the larger of x and y
public static int abs(int x);                 // Returns the absolute value of x
```

DrawingCanvas **class**:

```
public void clear();                                  // Clears the drawing canvas
public void setForeground(Color c);                   // Change the colour for later commands
public void drawLine(int x, int y, int u, int v);     // Draws line from (x, y) to (u, v)
public void drawRect(int x, int y, int wd, int ht);   // Draws outline of rectangle
public void fillRect(int x, int y, int wd, int ht);   // Draws solid rectangle
public void clearRect(int x, int y, int wd, int ht);  // Draws clear rectangle
public void drawOval(int x, int y, int wd, int ht);   // Draws outline of oval
public void fillOval(int x, int y, int wd, int ht);   // Draws solid oval
```

Color **class**:

```
public Color(int red, int green, int blue);   // Make a colour; arguments must be 0..255
Color.gray, Color.blue, Color.red,            // Some of the predefined colours
Color.green, Color.black, Color.white
```