Name: . . . .

ID Number: . . . .

Signature: . . . .

COMP102: Test

26 April, 2006

Instructions

- Time allowed: **90 minutes** ($1\frac{1}{2}$ hours).
- Answer **all** the questions.
- There are 85 marks in total.
- 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 25% of your final grade.
- Non-electronic translation dictionaries and calculators without a full set of alphabet keys are permitted.

Questions	**Marks**	
1. Basic Java	[22]	
2. Loops with Numbers	[20]	
3. Loops with Files	[8]	
4. Objects with Arrays of Objects	[21]	
5. Debugging Loops	[14]	
	TOTAL:	

Question 1. Basic Java [22 marks]

(a) [4 marks] For each of the four terms on the left below, draw a line from the term to the **best** matching definition on the right.

"variable"	A description of an action that can be performed on an object
	The name of a place in an object that holds a value indefinitely
"field"	A value passed to a method when the method is called
	A description of a kind of object, including its fields and its methods
"method"	A specification of the kind of value that a method returns
	The name of a place that holds a value, local to a method
"class"	A single instruction, inside a method

variable = The name of a place that holds a value, local to a method
field = The name of a place in an object that holds a value indefinitely
method = A description of an action that can be performed on an object
class = A description of a kind of object, including its fields and its methods

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

```
int num = 7;
System.out.println("A:  " + num);
System.out.println("B:  " + "num");
System.out.println("C:  " + (num + num));
System.out.println("D:  " + (num / 2));
```

```
A: 7
B: num
C: 14
D: 3
```

(Question 1 continued on next page)

(c) [2 marks] Write a fragment of Java that declares a variable called nm that can hold a value of type *String*, and puts the value "Hello" into the variable.

```
String nm = "Hello";
Or
String nm;
nm = "Hello";
```

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

```
double size = 1.1413;
size = size + 2;
System.out.println(size);
if ( size <= 25 ){
   size = size * 2.0;
}
else {
   size = Math.max(40.001, 45.497);
}
System.out.printf("Size = %4.2f\n", size);
```

```
3.1413
Size = 6.28
```

(e) [5 marks] Assume that the variable ans is of type *String*. Write a fragment of Java that will print (to the terminal window) the message "Yes" if ans contains the value "Dog" and the message "Sorry" if ans contains anything else.

```
if ("Dog".equals(ans)) { // Note, if ans is null, can't call equals on it
   System.out.println("Yes");
}
else {
   System.out.println("Sorry");
}
```

(Question 1 continued on next page)

(f) [4 marks] Assume that words is a variable of type *String*[ ] and contains an array of 100 Strings (with no null values).

Write a fragment of Java that will print all the strings in words to the terminal window, one per line.

```
    for (int i = 0; i < 100; i++){
        System.out.println(words[i]);
    }
OR
    for (int i = 0; i < words.length; i++){
        System.out.println(words[i]);
    }
OR
    int i = 0;
    while (i < 100){
        System.out.println(words[I]);
        i = i + 1;
    }
OR
    for (String wd : words){
        System.out.println(wd);
OR ...
```

Question 2. Loops with numbers [20 marks]

(a) [5 marks] What would the following timesTable method print out if it were called with the argument 5?

(For example, it might be called in the statement tbl.timesTable(5) where tbl is an object of the appropriate class.)

```
public void timesTable (int num){
   System.out.printf("Times table for %d: \n", num);
   int i = num;
   while (i > 0){
      i = i – 1;
      System.out.printf("%d x %d  =  %d \n", num, i, (num+i));
   }
   System.out.println("Done");
}
```

```
Times table for 5:
5 x 4 = 9
5 x 3 = 8
5 x 2 = 7
5 x 1 = 6
5 x 0 = 5
Done
```

The triangle method below is intended to print out a triangular table of numbers. For example, if triangle is called with the argument 5, it should print out the table on the right. However, triangle contains errors.

```
public void triangle (int max){
   int row = 0;
   while (row < max){
      int col = 0;
      while (col < max){
         col = col + 1;
         System.out.printf("%2d  ", row);
      }
      row = row + 1;
   }
}
```

```
1
2  1
3  2  1
4  3  2  1
5  4  3  2  1
```

(Question 2 continued on next page)

(Question 2 continued)

(b) [6 marks] What would the triangle method actually print out if it were called with the argument 5?

```
0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
```

(c) [9 marks] Write a correct version of the triangle method. If the argument is n, it should print n rows, where the ith row of the output should contain the integers from i down to 1.

```
public void triangle (int max){
    int row = 1;
    while (row <= max){
      int col = row;
      while (col > 0){
        System.out.printf("%d  ", col);
        col = col – 1;
      }
      System.out.println();
      row = row + 1;
    }
OR
  for (int row = 1; row <= max; row++){
    for (int col = row; col > 0; col– –){
        System.out.printf("%d  ", col);
    }
    System.out.println();
  }
OR
  for (int row = 1; row <= max; row++){
    for (int col = 0; col < row; col++){
        System.out.printf("%d  ", (row – col));
    }
    System.out.println();
  }
```

OR, even a single loop ...

```
    int row = 1;
    int col = 0;
    while (row <= max){
        System.out.printf("%d  ", (row – col));
        col = col + 1;
        if (col == row) {
            System.out.println();
            row = row + 1;
            col = 0;
    }
```

OR . . .   (there are many possible variations.)

}

Question 3. Loops with files [8 marks]

Complete the following printFile method.

printFile should print out the contents of a file, along with line numbers. Its argument is the name of the file. It should open the file using a Scanner, then read each line of the file and print (to the terminal window) the line number and the line. Finally, it should close the file.

For example, if the file contained the text shown on the left, printFile should print out the result on the right:

File

```
This is a small file with five
lines of text. The text would
cover 1700 square millimeters.
It has over 135 characters and
at least 27 words.
```

Output

```
1: This is a small file with five
2: lines of text. The text would
3: cover 1700 square millimeters.
4: It has over 135 characters and
5: at least 27 words.
```

Reminder: there is documentation on the Scanner class at the end of the test.

```
public void printFile(String fname){
   try{
      Scanner sc = new Scanner(new File(fname));
      int lineNum = 1;
      while (sc.hasNext()){
         String line = sc.nextLine();
         System.out.println(lineNum + ":  "+ line);
         lineNum++;
      }
      sc.close();
   }
   catch(Exception e){System.out.printf("Printing file %s failed\n", fname);}
}
```

Question 4. Objects with Arrays of Objects [21 marks]

This question concerns a program for the University Safety officer who needs to keep track of floor wardens in university buildings. Each floor may have one warden assigned to it, or it may have no warden.

The program has several classes, including Building and Warden. The Warden class is given below. A Warden object has fields to store the name and contact information for the warden.

An outline of the Building class is given on the facing page. A Building object has fields to store the name of the building, the number of floors in the building, and an array of the floor wardens. In the array, the ith cell contains the warden assigned to the ith floor. A **null** in the ith cell means that no warden is currently assigned to the ith floor. Note, the floors of a building are numbered from 0.

(a) [8 marks] On the facing page, complete the constructor and the getName, setWarden and getWarden methods for the Building class:

- The constructor should assign values to the fields of the *Building* object, including a new, empty array of the appropriate size.
- getName should return the name of the building.
- setWarden has two parameters: a floor number and a Warden object. It should put the Warden object in the appropriate cell of the array.
- getWarden should return the Warden of the specified floor of the building. If there is no Warden for that floor, it returns **null**.

```
public class Warden{
   private String name;
   private String phone;

   /** Construct a new Warden object */
   public Warden(String name, String phone){
      this.name = name;
      this.phone = phone;
   }

   /** Return a String containing name and phone number. */
   public String toString(){
      return this.name + " ph "+ this.phone;
   }

}
```

(Question 4 continued on next page)

```
public class Building{
   private String name;              // name of building
   private int numFloors;            // number of floors in building
   private Warden[ ] wardens;        // array of Warden (or null) on each floor.

   /** Construct a new Building object */
   public Building(String name, int numFloors){
      this.name = name;
      this.numFloors = numFloors;
      this.wardens = new Warden[numFloors];
   }

   /** Return the name of the building */
   public String getName(){
      return this.name;
   }

   /** Set the warden on the given floor */
   public void setWarden(int floor, Warden w){
      this.wardens[floor] = w;
   }

   /** Return the warden on the given floor */
   public Warden getWarden(int floor){
     return this.wardens[floor];
   }

   public void print(){ . . .                       // to be completed on next page
   public Warden closestWarden(int fl){ . . .       // to be completed on next page
}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [5 marks] Complete the following print method for the Building class.

print should print (to the terminal window) the name of the building, followed by the wardens for each floor. It should not print anything for floors that do not have a warden.

For example, if the building is called "Hunter" and has 6 floors, and there are wardens for floors 0, 1, and 5, but none for floors 2 to 4, then the method might print out the following:

```
Wardens for Hunter:
 Floor 0: Pondy ph x1234
 Floor 1: Lindsay ph x4321
 Floor 5: John ph x2134
```

```
/** Print out building and all the wardens */
public void print(){
    System.out.printf("Wardens for %s:\n", this.name);
    for (int fl = 0; fl < this.numFloors; fl++){
        if (this.wardens[fl]  !=  null){
            System.out.printf(" Floor %d: %s\n", fl, this.wardens[fl].toString());
        }
    }
Note: the toString() is not actually necessary, since Java automatically calls
the toString() method on an object to coerce it to a string, eg to + it to another
string or to print it using println or printf(...%s )!
}
```

(c) [8 marks] (Harder) Complete the closestWarden method for the Building class on the facing page.

closestWarden has a floor number as a parameter, and should return the warden assigned to the floor closest to the specified floor. This will be the warden assigned to the specified floor, if there is one; otherwise, it will be the warden on the closest floor that has a warden. If the closest warden above the specified floor is the same distance as the closest warden below the specified floor, the method should return the one on the higher floor. It should return **null** if there are no wardens on any floor.

For the "Hunter" example above, closestWarden(2), should return the warden on floor 1, and closestWarden(3) should return the warden on floor 5.

(Question 4 continued on next page)

```
/** Find and return the warden closest to the specified floor. */
public Warden closestWarden(int fl){
 //The first two versions search from fl outwards (up and down together)
  int upper = fl;
  int lower = fl;
  while(upper < this.numFloors || lower >= 0){
    if (upper < this.numFloors){
      if (this.wardens[upper] != null){
        return this.wardens[upper];
      }
      upper++;
    }
    if (lower >= 0){
      if (this.wardens[lower] != null){
        return this.wardens[lower];
      }
      lower--;
    }
  }
  return null;

 OR

   for (int dist = 0; dist < this.numFloors; dist++){
     if ( fl+dist < this.numFloors && this.wardens[fl+dist] != null)
       return this.wardens[fl+dist];
     if ( fl–dist >= 0 && this.wardens[fl–dist] != null)
       return this.wardens[fl–dist];
   }
   return null; // if there were no wardens at all
}

 OR (see next page)
```

```
public Warden closestWarden(int fl){
  // This version searches for the closest above then the closest below
  // Then works out which was closest
    int upper = fl;
    int lower = fl;
    while (upper < this.numFloors){
        if (this.wardens[upper] != null) break;
        upper++;
    }
    while (lower >= 0){
        if (this.wardens[lower] != null) break;
        lower--;
    }
    if (upper == this.numFloors && lower == 0)
        return null;
    else if (upper–fl <= fl–lower)
        return this.wardens[upper];
    else
        return this.wardens[lower];
}
```

OR (there are lots of possible variants)

Question 5. Debugging Loops (Harder) [14 marks]

The largestNum method below is intended to find and return the largest number in a file. Its argument is a Scanner object that it assumes is already connected to an open file. The file may contain a mixture of words and integers. For example, given the following file, largestNum should return the value 28.

```
This file has 28
letters and 2 numbers
```

The method compiles correctly, but has several logical errors.

```
public int largestNum(Scanner scan){
   int max = 0;
   while(scan.hasNext()){
      if (scan.hasNextInt()){
         if (scan.nextInt() > max){
            max = scan.nextInt();
         }
      }
   }
   return max;
}
```

(a) [4 marks] What would largestNum return if it were called on the following file of integers?

```
 -50 -4 1500 100
 150 135 128
```

135

(b) [3 marks] Why would largestNum never return a value if it were called on the following file?

```
This file has 28
letters and 2 numbers
```

It would loop for ever because the first token is not an integer
so it never reads the the first token, and keeps checking it
for ever.

(Question 5 continued on next page)

(c) [2 marks] Briefly describe one other error in the largestNum method.

max should be initialised to Integer.MIN_VALUE, not 0
If all the numbers in the file are negative, it will not
find the largest number - it will just return 0.
OR
When it finds that an integer is larger than max,
it then assigns the *following* integer to max
so that it will remember the wrong number.

(d) [5 marks] Write a correct version of the largestNum method.

```
public int largestNum(Scanner file){
// Assumes file is already connected to a file
        int max = Integer.MIN_VALUE;
        while(file.hasNext()){
         if (file.hasNextInt()){
             int n = file.nextInt();
             if (n > max){
                 max = n;
             }
         }
         else
             file.next();
        }
    return max;
}
```

****

Brief, incomplete 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);	*Print* s *with no newline*
public *void* print(*int* i);	*Print* i *with no newline*
public *void* print(*double* d);	*Print* d *with no newline*
public *void* println();	*Print a newline*
public *void* println(*String* s);	*Print* s *followed by newline*
public *void* println(*int* i);	*Print* i *followed by newline*
public *void* println(*double* d);	*Print* d *followed by newline*
public *void* printf(*String* format, ...);	*Print the* format *string, inserting the remaining arguments at the* %*'s in the format string:* %3d *for* int*s, (using at least 3 characters),* %4.2f *for* double*s (4 characters and 2 decimal places),* %s *for* String*s.* *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 *boolean* hasNext();	*Return true if there is more to read*
public *boolean* hasNextInt();	*Return true if the next token is an integer*
public *boolean* hasNextDouble();	*Return true if the next token is a number*
public *String* next();	*Return the next token (word)*
public *String* nextLine();	*Return the next line*
public *int* nextInt();	*Return the integer value of the next token* *(throws exception is next token is not an integer)*
public *double* nextDouble();	*Return the double value of the next token* *(throws exception is next token is not a number)*
public *void* close();	*Close 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})
public static *int* parseInt(*String* str);	*Return the integer represented by the string*

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 *boolean* startsWith(*String* s);	*First part of string matches* s
public *boolean* contains(*String* s);	s *matches some part of the string*
public *int* indexOf(*String* s);	*Returns -1 if it does not contain* s *anywhere otherwise, returns the index of where* s *first matches*

Math class:

public static *double* min(*double* x, *double* y);	*Return the smaller of* x *and* y
public static *double* max(*double* x, *double* y);	*Return the larger of* x *and* y
public static *double* abs(*double* x);	*Return the absolute value of* x
public static *int* min(*int* x, *int* y);	*Return the smaller of* x *and* y
public static *int* max(*int* x, *int* y);	*Return the larger of* x *and* y
public static *int* abs(*int* x);	*Return 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);	*Draw line from* (x, y) *to* (u, v)
public *void* drawRect(*int* x, *int* y, *int* wd, *int* ht);	*Draw outline of rectangle*
public *void* fillRect(*int* x, *int* y, *int* wd, *int* ht);	*Draw solid rectangle*
public *void* clearRect(*int* x, *int* y, *int* wd, *int* ht);	*Draw clear rectangle*
public *void* drawOval(*int* x, *int* y, *int* wd, *int* ht);	*Draw outline of oval*
public *void* fillOval(*int* x, *int* y, *int* wd, *int* ht);	*Draw solid oval*