

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]	<input type="checkbox"/>
2. Loops with Numbers	[20]	<input type="checkbox"/>
3. Loops with Files	[8]	<input type="checkbox"/>
4. Objects with Arrays of Objects	[21]	<input type="checkbox"/>
5. Debugging Loops	[14]	<input type="checkbox"/>
	TOTAL:	<input type="checkbox"/>

**Please answer the following question.**

(Your answer will not affect your mark in any way.)

How much of the content in the course so far had you learned before you started this course?

None	Some	Lots
------	------	------

**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

[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

(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 ) );
```

(Question 1 continued on next page)

(Question 1 continued)

(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.

(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);
```

(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.

(Question 1 continued on next page)

**(Question 1 continued)**

**(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.

## 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");
}
```

(Question 2 continued on next page)

**(Question 2 continued)**

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
```

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

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

```
public void triangle (int max){

}
}
```

**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 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 =
        int lineNum =

    }
    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 *i*th cell contains the warden assigned to the *i*th floor. A **null** in the *i*th cell means that no warden is currently assigned to the *i*th 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 nameAndContact() {
        return this.name + " ph " + this.phone;
    }
}
```

(Question 4 continued on next page)

(Question 4 continued)

```
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;

    }

    /** Return the name of the building */
    public String getName(){

    }

    /** Set the warden on the given floor */
    public void setWarden(int floor, Warden w){

    }

    /** Return the warden on the given floor */
    public Warden getWarden(int 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)**

(c) [8 marks] (Harder) Complete the following `closestWarden` method for the `Building` class.

`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 on the facing page, `closestWarden(2)`, should return the warden on floor 1, and `closestWarden(3)` should return the warden on floor 5.

```
/** Find and return the warden closest to the specified floor. */  
public Warden closestWarden(int fl){
```

```
}
```

### 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
```

(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
```

(Question 5 continued on next page)



## Brief, incomplete documentation of some classes and methods

### PrintStream class:

*Note, System.out is a PrintStream object*

<b>public</b> PrintStream( <i>File</i> f);	<i>Constructor, for printing to a file</i>
<b>public</b> void <u>close</u> ();	<i>Close the file (if it is wrapping a File object)</i>
<b>public</b> void <u>print</u> ( <i>String</i> s);	<i>Print s with no newline</i>
<b>public</b> void <u>print</u> ( <i>int</i> i);	<i>Print i with no newline</i>
<b>public</b> void <u>print</u> ( <i>double</i> d);	<i>Print d with no newline</i>
<b>public</b> void <u>println</u> ();	<i>Print a newline</i>
<b>public</b> void <u>println</u> ( <i>String</i> s);	<i>Print s followed by newline</i>
<b>public</b> void <u>println</u> ( <i>int</i> i);	<i>Print i followed by newline</i>
<b>public</b> void <u>println</u> ( <i>double</i> d);	<i>Print d followed by newline</i>
<b>public</b> void <u>printf</u> ( <i>String</i> format, ...);	<i>Print the format string, inserting the remaining arguments at the %'s in the format string: %3d for ints, (using at least 3 characters), %4.2f for doubles (4 characters and 2 decimal places), %s for Strings. Use \n for newline</i>

### Scanner class:

<b>public</b> Scanner( <i>InputStream</i> i);	<i>Constructor. Note System.in is an InputStream</i>
<b>public</b> Scanner( <i>File</i> f);	<i>Constructor, for reading from a file</i>
<b>public</b> boolean <u>hasNext</u> ();	<i>Return true if there is more to read</i>
<b>public</b> boolean <u>hasNextInt</u> ();	<i>Return true if the next token is an integer</i>
<b>public</b> boolean <u>hasNextDouble</u> ();	<i>Return true if the next token is a number</i>
<b>public</b> <i>String</i> <u>next</u> ();	<i>Return the next token (word)</i>
<b>public</b> <i>String</i> <u>nextLine</u> ();	<i>Return the next line</i>
<b>public</b> <i>int</i> <u>nextInt</u> ();	<i>Return the integer value of the next token (throws exception is next token is not an integer)</i>
<b>public</b> <i>double</i> <u>nextDouble</u> ();	<i>Return the double value of the next token (throws exception is next token is not a number)</i>
<b>public</b> void <u>close</u> ();	<i>Close the file (if it is wrapping a File object)</i>

**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*

**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.