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

ID Number: ..... **Signature** .....

Model Solutions
COMP 102: Test 1

31 March, 2015

Instructions

- Time allowed: **50 minutes** .
- Answer **all** the questions. There are 45 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.
- Brief Java documentation is provided with the test
- This test contributes 15% of your final grade
  (But your mark will be boosted up to your exam mark if that is higher.)
- You may use paper translation dictionaries, and calculators without a full set of alphabet keys.
- You may write notes and working on this paper, but make sure your answers are clear.

Questions	Marks	
1. Understanding programs	[8]	
2. Writing programs with input and output	[7]	
3. Writing methods that use objects	[6]	
4. Understanding arguments and parameters	[8]	
5. Defining and calling methods	[8]	
6. Writing methods with **if** and **while**	[8]	
	TOTAL:	

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. Understanding programs [8 marks]

(a) [4 marks] Understanding variables

What will the following method print out?

```
public void printStuff (){
        int x = 5;
        int y = 10;
        UI.println("Line 1: " + y);
        UI.println("Line 2: " + (x+1));
        x = x + 2;
        y = y + x;
        UI.println("Line 3: "+ x);
        UI.println("Line 4: "+ y);
}
```

x:

y:

```
Line 1:  10
Line 2:  6
Line 3:  7
Line 4:  17
```

(Question 1 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

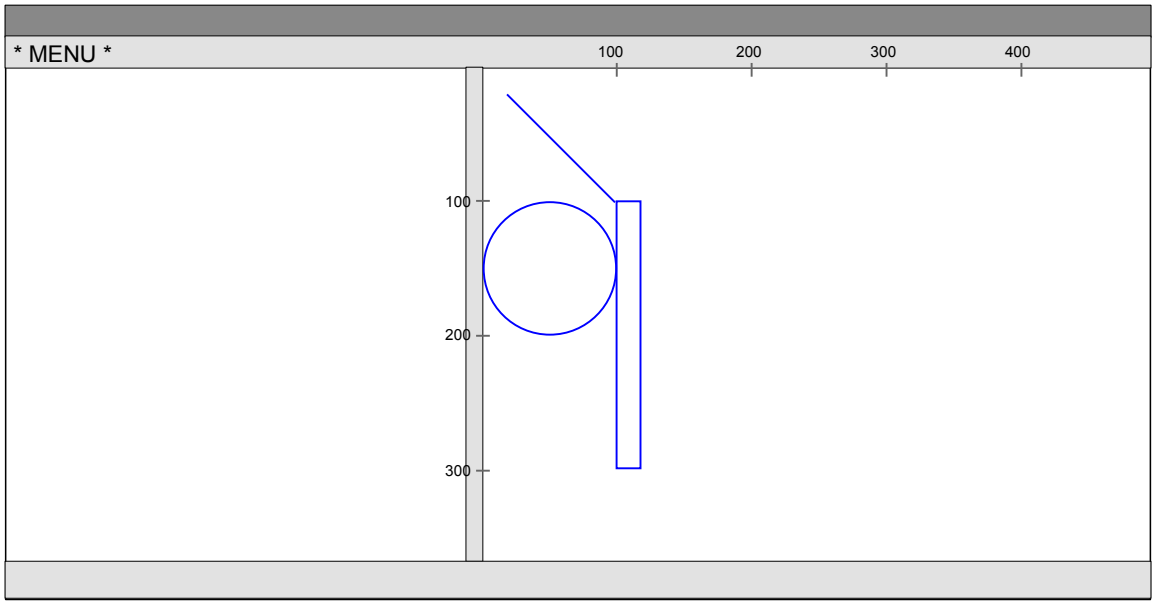
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 1 continued)

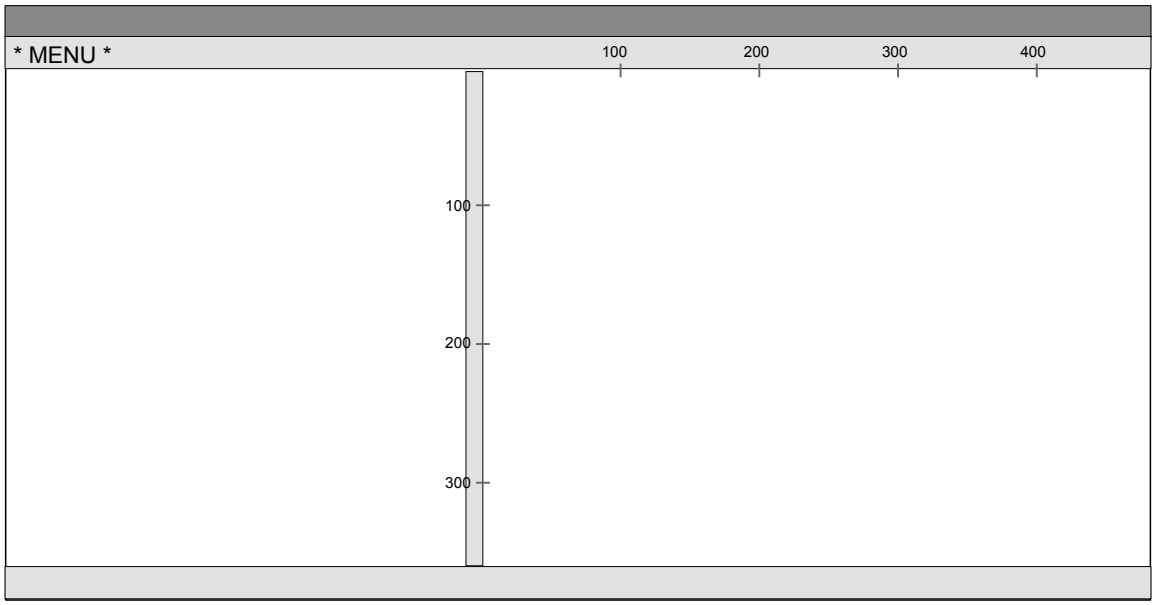
(b) [4 marks] Understanding graphical output

Sketch what the following method will draw in the graphics pane if the user enters the value 100?

```
public void drawStuff(){
        int ans = UI.askInt("position: ");
        UI.drawLine(20, 20, ans, ans);
        UI.drawRect(ans, ans, 20, ans*2);
        UI.drawOval(0, ans, ans, ans);
}
```



Extra copy:



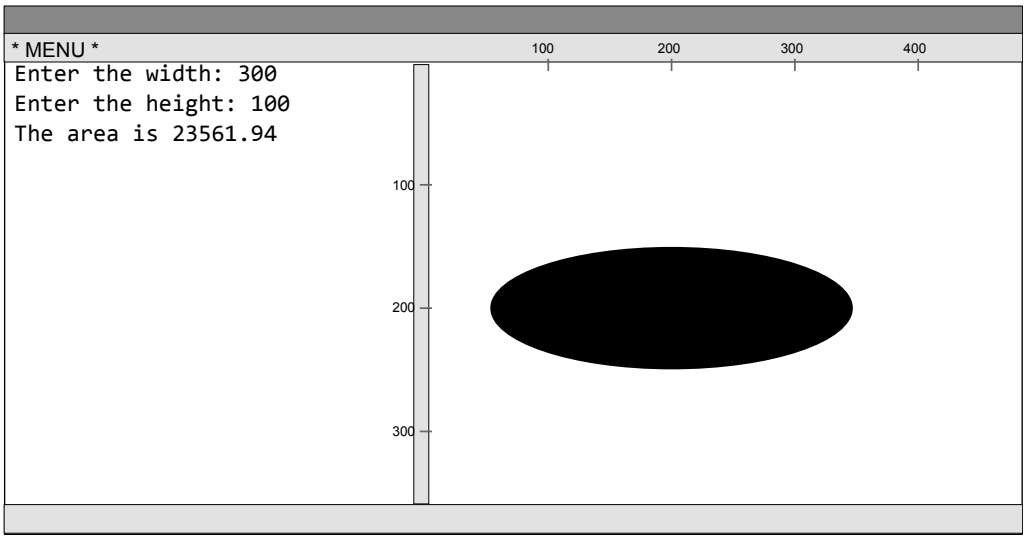
SPARE PAGE FOR EXTRA ANSWERS

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

Question 2. Writing programs with input and output [7 marks]

Complete the showOval method below to draw an oval in the graphics pane and print out its area. The method should ask the user for the width and the height of the oval, calculate and print out the area, and draw a filled oval centered at (200, 200).

For example, if you called showOval and entered 300 for the width and 100 for the height, the text pane and graphics pane should look like this:



Hints:

- The area of an oval is $width \times height \times \pi/4$,
- Math.PI is a predefined constant with the value of π

```
public void showOval() {
        UI.clearText();              // clears the text pane
        UI.clearGraphics();          // clears the graphics pane

        double width = UI.askDouble("Enter the width:");
        double height = UI.askDouble("Enter the height:");
        double area = width * height * Math.PI /4;
        UI.println("The area is " + area);

        double left = 200 - width/2;
        double top = 200 - height/2;
        UI.fillOval(left, top, width, height);




}
```

Question 3. Writing methods that use objects [6 marks]

Suppose the GameGun class has one constructor and two methods:

Constructor:

```
public GameGun(double x, double y, double size, Color c)
// constructs a GameGun object, with the specified position, size and color,
// displays it at the specified position on the graphics pane.
```

Methods:

```
public void move(String direction, int dist)
// makes the gun move in the direction specified in the parameter
// for the specified distance
// Directions can be "North", "South", "East", or "West"

public void fire (int rounds)
// makes the gun fire the specified number of rounds.
```

Complete the following game method, which should first create two GameGun objects at different positions, with different sizes and different colors. It should then make the first gun move south 50 units and fire 20 rounds, then make the second gun move east 100 units and fire 15 rounds, and finally move the first gun west 20 units and then fire 5 rounds from each gun.

```
public void game(){
        GameGun g1 = new GameGun(100, 30, 20, Color.red);
        GameGun g2 = new GameGun(200, 50, 50, Color.blue);
        g1.move("South", 50);
        g1. fire (20);
        g2.move("East", 100);
        g2. fire (15);
        g1.move("West",20);
        g2. fire (5);
        g1. fire (5);









}
```

Question 4. Understanding Arguments and Parameters [8 marks]

The following listStaff method calls the printPerson method several times. Show what listStaff will print in the text pane.

```
public void listStaff (){
    String expertise = "Databases";
    String phone = "1234";
    UI.println("Current library staff:");
    this.printPerson("Jack", "4321", "Reference");
    this.printPerson("Jim", phone, expertise);
    String name = "Jane";
    expertise = "Restocking";
    this.printPerson(expertise, "3535", name);
    this.printPerson("Julia", expertise, phone);
}
```

expertise: _____

phone: _____

name: _____

```
public void printPerson(String name, String phone, String exp){
    UI.println(name + " (" + exp + ")");
    UI.println(" phone: " + phone);
    UI.println("---");
}
```

name: _____

phone: _____

exp: _____

```
Current library staff:
Jack (Reference)
phone:  4321
---
Jim (Databases)
phone:  1234
---
Restocking (Jane)
phone:  3535
---
Julia (1234)
phone:  Restocking
---
```

Question 5. Defining and Calling Methods [8 marks]

The following shoppingList method prints out labels for a number of shopping items and then prints the total cost of the items at the end. The labels show the name, the quantity, the unit price of the item, and the cost (quantity $\times$ unit price) of the item.

```
public void shoppingList(){
        UI.println("+------------------------------");
        UI.println("| name: Bread");
        UI.println("| quantity: 1");
        UI.println("| unit price: $3.50");
        UI.println("| cost: $3.50");
        UI.println("+------------------------------");
        UI.println("+------------------------------");
        UI.println("| name: milk");
        UI.println("| quantity: 2");
        UI.println("| unit price: $4.20");
        UI.println("| cost: $8.40");
        UI.println("+------------------------------");
        UI.println("+------------------------------");
        UI.println("| name: beer");
        UI.println("| quantity: 4");
        UI.println("| unit price: $3.00");
        UI.println("| cost: $12.00");
        UI.println("+------------------------------");
        UI.println("+------------------------------");
        UI.println("| name: chocolate");
        UI.println("| quantity: 1");
        UI.println("| unit price: $4.80");
        UI.println("| cost: $4.80");
        UI.println("+------------------------------");
        UI.println();
        UI.println("Total cost is: $28.70");
    }
```

shoppinglist is not well designed: it has lots of repetition, and required the programmer to do lots of calculations. It would be better design to define another method called labelItem which calculates the cost of an item from the quantity and the unit price and then prints out a single label for the itme; and then make shoppinglist call the labelItem method four times and compute the total.

Complete the definitions of shoppinglistFixed and labelItem on the facing page so that they print out the same labels as the original version. You will need to determine

- the appropriate arguments for the calls to labelItem,
- the appropriate header of labelItem, and
- the statements in the body of labelItem.

Note that labelItem should return the cost of the item. You may assume that the quantity is always an integer.

```
public void shoppinglistFixed (){

        double cost = this.labelItem(    "bread", 1, 3.5);

        cost = cost + this.labelItem(    "milk", 2, 4.2);

        cost = cost + this.labelItem(    "beer", 4, 3.0);

        cost = cost + this.labelItem(    "chocolate", 1, 4.8);

        UI.println ();
        UI.printf ("Total cost is: $%4.2f\n", cost);
}



public      double      labelItem(String name, int quantity, double price){

         double cost = price * quantity;
        UI.println ("--------------------------------");
         UI.println ("| name: " + name);
         UI.println ("| quantity: " + quantity);
         UI.printf ("| unit price: $%4.2f\n", price);
         UI.printf ("| cost: $%4.2f\n", cost);









        UI.println ("--------------------------------");

        return                      ;

}
```

Question 6. Writing methods with if and while. [Hard] [8 marks]

Complete the following checkSentence method which should check whether the given String is an acceptable sentence for a particular web form. It should return false if the string fails any of the conditions, and should return true if the string passes all the conditions.

An acceptable sentence

1. must end with the full stop character "."
2. must start with a capital letter from "A" to "Z".
3. must not contain a full stop character "." anywhere in the sentence except the end.
4. must not contain more than 50 words (where words are separated by spaces).

The more conditions you check, the higher your mark!

Hint: Look at the methods for the String class on the second page of the documentation.

(Question 6 continued on next page)

(Question 6 continued)

```
public boolean checkSentence(String sentence){

    int len = sentence.length();
    if (len < 2) {return false;}

    String last = sentence.substring(len-1,len);
    if (!last.equals(".")){ return false; }

    String first = sentence.substring(0,1);
    if ( first.compareTo("A") < 0 || first.compareTo("Z") > 0 ) {
        return false;
    }
    int index = sentence.indexOf(".");
    if (index > -1 && index < len-1) {return false;}

    int count = 0;
    String rest = sentence.trim();
    while (rest.indexOf(" ")>0){
        count++;
        rest = rest.substring(rest.indexOf(" "), rest.length()).trim();
    }
    return (count <=50);









}
```

* * * * * * * * * * * * * * *

Alternative ways of checking the constraints for question 6:

```
//ends with a .
        if (! sentence.endsWith(".")) { return false; }

//does not contain . before the end.
        sentence = sentence.substring(0, sentence.length()-1);
        if (sentence.contains(".")) { return false; }

// . only at the end (both conditions at once)
        if (sentence.indexOf(".")!=sentence.length()-1) { return false; }


// Starts with capital: first letter is one of the capital letters:
       if (! ("ABCDEFGHIJKLMNOPQRSTUVWXYZ".contains(sentence.substring(0,1)))){
          return false;
       }
// Starts with capital: first letter is equal to its upperCase version, but have
// to check that it is not punctuation or a digit
        if (!sentence.substring(0,1).equals(sentence.substring(0,1).toUpperCase())
            || sentence.substring(0,1).equals(sentence.substring(0,1).toLowerCase())){
          return false;
        }
// Starts with capital: sentence isn't before A and is before "[" (first character after "Z")
        if (sentence.compareTo("A") < 0 || sentence.compareTo("[")>= 0 ){return false;}

//Count words: (Note: we hadn't covered scanners by the time of the test!)
        int count = 0;
        Scanner sc = new Scanner(sentence);;
        while (sc.hasNext()){
            sc.next();
            count++;
        }
        return (count <=50);
```