

COMP1500/COMP7901

Lab Assessment

Second Semester 2004

Lab Assessment 1

Before attempting this assessment you will need to complete your study of Chapters 1 and 2 of *Java Genesis*.

PROBLEM: constructing a solid blue square

Construct a class named `DrawSolidSquare` by modifying the class `DrawSquare` (see Problem 10 in Chapter 2 of *Java Genesis*) so that when the class is compiled and run, a blue solid square with width 70 pixels and with its centre at the position (200, 100) is displayed in a window, i.e. the x-coordinate of the centre of the square is 200 and the y-coordinate is 100, measured in pixels from the upper-left corner of the window's display region.



To earn your 2 marks for Lab Assessment 1, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 5 (the week beginning 23rd August)**.

Lab Assessment 2

Before attempting this assessment you will need to complete your study of Chapter 3 of *Java Genesis*.

PROBLEM: growing a square

In the spirit of Problem 14 *animating the moving square* in Chapter 3 of *Java Genesis*, construct a class `GrowSquare` that uses the statement



```
SquareFigure.create();
```

to first create a red square of width 50 pixels in the centre of a window. The square then grows, with its width increasing by 2 pixels every 50 milliseconds until its width is 250 pixels. At all times the square remains at the centre of the window.

Consult Section 2.4 of *Java Genesis* to recall the various messages that can be sent to the `SquareFigure` class.

□

To earn your 2 marks for Lab Assessment 2, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 5 (the week beginning 23rd August)**.

Lab Assessment 3

Before attempting this assessment you will need to complete your study of Chapter 4 of *Java Genesis*.



PROBLEM: bouncing the circle

Construct a class `BounceCircle` which when compiled and run uses the statement

```
CircleFigure.create();
```

to first create a red circle of radius 25 pixels in the centre of a window. The circle then begins moving to the left at the speed of 1 pixel every 20 milliseconds. When it reaches the left edge of the window (i.e. when the left edge of the circle is at $x = 0$) it bounces off this edge and moves right at the same speed until it bounces off the right edge of the window (i.e. when the right edge of the circle is at $x = 392$) and moves left again at the same speed. This bouncing off the left and right edges of the window continues indefinitely until the program is terminated (e.g. by clicking with the mouse on the 'X' in the upper-right corner of the window).

Consult *Java Genesis* Section 2.4 and Problems 3 and 13 of Chapter 4 to recall the various messages that can be sent to the `CircleFigure` class.

□

To earn your 2 marks for Lab Assessment 3, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 5 (the week beginning 23rd August)**.

Lab Assessment 4

Before attempting this assessment you will need to complete your study of Chapters 5 and 6 of *Java Genesis*.

PROBLEM: searching an array



This problem was previously set as an exam question. Consider the following incomplete code for the class `Marks` (the code can be found in the folder `genesis\projects\Lab Assessment\Searching An Array`):

```
import genesis.*;

public class Marks {

    public static void main (String [ ] args) {
        int [ ] marks = {57, 91, 72, 84, 63, 45, 95, 66, 77, 89};
        Transcript.println(numbersInRange(marks, 60, 80));
        Transcript.println(nearestToTarget(93, marks));
    }

    public static int numbersInRange
        (int [ ] intArray, int lower, int upper) {
        // ...
        // complete the code for this method
        // ...
    }

    public static int nearestToTarget
        (int target, int [ ] intArray) {
        // ...
        // complete the code for this method
        // ...
    }
}
```

Complete the code for the methods `numbersInRange` and `nearestToTarget`.

The method `numbersInRange` has three parameters. The first parameter denotes an array of integers; the second and third parameters are integers denoting a lower and upper limit respectively. The method returns an integer giving the number of elements in the array whose value is strictly greater than the value of the second parameter but strictly less than the value of the third parameter.

The method `nearestToTarget` has two parameters. The first denotes a 'target'; the second an array of integers. The method returns the value of the integer in the array closest to the target. If there is more than one such integer in the array, the first one in the array is returned. You may assume the array is not empty.

Note: If the class `Marks` is compiled and run after the two methods have been coded, you should get the numbers 4 and 91 printed on separate lines in the Transcript window.

□

To earn your 2 marks for Lab Assessment 4, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 8 (the week beginning 13th September)**.

Lab Assessment 5

Before attempting this assessment you will need to complete your study of Chapter 7 of *Java Genesis*.



PROBLEM: strange polygon

Revisit Problem 6 *rotating a regular polygon* of Chapter 7 of *Java Genesis*. The code for the class `RotatingPolygon` can be found in the `genesis\projects\Lab Assessment\Strange Polygon` folder. When this class is run a dialog box opens requesting the number of edges. A regular polygon with the given number of edges is then drawn and rotates clockwise about its centre. This is achieved by rotating each vertex of the polygon clockwise about the centre by 0.01 radians every 5 milliseconds.

Modify the code for the class `RotatingPolygon` to construct a new class called `StrangePolygon` which when compiled and run opens a dialog box requesting the number of edges. A regular polygon with the given number of edges is then drawn but this time the even-numbered vertices (i.e. vertices 0, 2, 4, etc.) are rotated clockwise about the centre by 0.01 radians every 20 milliseconds, while the remaining odd-numbered vertices are rotated anti-clockwise at the same speed.

□

To earn your 2 marks for Lab Assessment 5, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 8 (the week beginning 13th September)**.

Lab Assessment 6

Before attempting this assessment you will need to complete your study of Chapter 8 of *Java Genesis*.

PROBLEM: taxi or limousine



This problem was previously set as an exam question.

Consider the class `TaxiTrip` whose code is given below (the code for the classes for this problem can be found in the *genesis/projects/Lab Assessment/Taxi Or Limousine* folder):

```
public class TaxiTrip {

    private double hireFee = 4.50; // $'s to hire taxi
    private double rate = 1.25; // $'s per kilometre
    private double distanceTravelled;// kilometres travelled

    public TaxiTrip (double distance) {
        distanceTravelled = distance;
    }

    public double getFare ( ) {
        return
            (int)((hireFee + distanceTravelled*rate)*100)/100.0;
    }

    public String toString () {
        return "hire fee: "+hireFee+"\n"
            +"rate per kilometre: "+rate+"\n"
            +"distance travelled: "+distanceTravelled+"\n"
            +"fare owing: "+getFare();
    }

    public void setRate (double newRate) {
        rate = newRate;
    }

    public double getRate ( ) {
        return rate;
    }
}
```

An object of the class `TaxiTrip` denotes a journey undertaken in a taxi. Given the distance travelled, the fare owing is calculated by the method `getFare`. This fare consists of a fixed hire charge of \$4.50 plus \$1.25 for each kilometer travelled. The method `getFare` returns the fare truncated to two places of decimal.

To test this class, here is the code for a class `TestTaxi`:

```
import genesis.*;

public class TestTaxi {

    public static void main (String [] args) {
        TaxiTrip trip = new TaxiTrip(12.25);
        Transcript.println(trip);
    }
}
```

If you place both of the above classes in the same folder, compile them and run the class `TestTaxi`, the following output will appear in the Transcript window:

```
hire fee: $4.5
rate per kilometre: $1.25
distance travelled: 12.25
fare owing: $19.81
```

The assessment task

Your task is to code a class `LimoTrip` that is a subclass of `TaxiTrip`. The class `TaxiTrip` itself must not be changed.

An object of the class `LimoTrip` denotes a journey undertaken in a limousine. The hire fee for a limousine is the same as for a taxi, but the rate per kilometer for a limousine is double the rate per kilometer currently being charged for a taxi. In addition, an extra charge of 75 cents for each passenger is added to the fare.

To help you, at the top of the next page we have given incomplete code for the class `LimoTrip`, together with complete code for a class `TestLimo`. After you have completed coding your class `LimoTrip`, compile the classes `LimoTrip` and `TestLimo` and run the class `TestLimo`.

```
public class LimoTrip extends TaxiTrip {

    private int numberPassengers; // number of passengers
    private double passengerCharge = 0.75;//charge per passenger

    public LimoTrip (int numPassengers, double distance) {
        // ...
        // complete the code for this constructor
        // ...
    }

    public double getFare ( ) {
        // ...
        // redefine this method
        // ...
    }

    public String toString () {
        // ...
        // redefine this method
        // ...
    }
}
```

```
import genesis.*;

public class TestLimo {

    public static void main (String [] args) {
        LimoTrip trip = new LimoTrip(6, 12.25);
        Transcript.println(trip);
    }
}
```

If you have coded the class `LimoTrip` correctly, when the class `TestLimo` is run the following output should appear in the Transcript window:

```
number of passengers: 6
charge per passenger: $0.75
hire fee: $4.5
rate per kilometre: $2.5
distance travelled: 12.25
fare owing: $39.62
```

Notice that the `toString` method in the class `LimoTrip` first supplies information about the number of passengers and the charge per passenger, and then supplies the same information as the `toString` method in the superclass `TaxiTrip`.

□

To earn your 2 marks for Lab Assessment 6, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 10 (the week beginning 4th October)**.

Lab Assessment 7

Before attempting this assessment you will need to complete your study of Chapter 9 of *Java Genesis*.



PROBLEM: moving with the mouse

This problem is similar to one previously set as an exam question.

Consider the the classes `MoveFrame`, `MovePanel` and `Main` whose code is given below (the code for the classes for this problem can be found in the folder *genesis\projects\Lab Assessment\Moving With The Mouse*). If these three classes are compiled and the `Main` class run, a window opens displaying the text `Java is cool`. Nothing happens when the mouse is moved or clicked inside the window.

```
import genesis.*;
import java.awt.*;

public class MoveFrame extends JFrame {

    public MoveFrame ( ) {
        setTitle("Follow the Mouse");
        setBounds(50, 100, 400, 400);
        Container c = getContentPane();
        MovePanel p = new MovePanel();
        c.add(p);
    }
}
```

```
import javax.swing.*;
import java.awt.*;

public class MovePanel extends JPanel {

    private int xPosn = 100, yPosn = 200;

    public MovePanel ( ) {
        setBackground(Color.white);
    }

    public void paintComponent (Graphics g) {
        super.paintComponent(g);
        g.setFont(new Font("SansSerif", Font.BOLD, 32));
        g.setColor(Color.red);
        g.drawString("Java is cool", xPosn, yPosn);
    }
}

public class Main {

    public static void main (String [] args) {
        MoveFrame win = new MoveFrame();
        win.setVisible(true);
    }
}
```

The assessment task

Modify the code for the class `MovePanel` so that when the mouse is moved within the window the text moves and follows the mouse.

Further modify the code for the class `MovePanel` so that when the mouse is pressed the text disappears and is replaced by a hollow blue square of width 100 pixels. The square moves with the mouse so that its centre is always at the mouse's current position. If the mouse is pressed again the text re-appears. Pressing the mouse toggles the display between the text and the square.



To earn your 2 marks for Lab Assessment 7, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 10 (the week beginning 4th October)**.

Lab Assessment 8

Before attempting this assessment you will need to complete your study of Chapter 10 of *Java Genesis*.



PROBLEM: a GUI for a circle

This problem is similar to one previously set as an exam question.

Consider the the classes `CircleGuiFrame`, `CircleGuiPanel` and `Main` whose code is given below (the code for the classes for this problem can be found in the folder *genesis\projects\Lab Assessment\GUI For A Circle*). If these three classes are compiled and the `Main` class run, a window opens displaying a solid circle of diameter 200 pixels at its centre. At the top of the window is a text field, and at the bottom is an action button 'draw'. Nothing happens when the action button is pressed.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import genesis.*;

public class CircleGuiFrame extends JFrame {

    private JTextField tf;
    private CircleGuiPanel p = new CircleGuiPanel();

    public CircleGuiFrame ( ) {
        setTitle("Circle GUI");
        setBounds(100, 100, 400, 400);
        tf = new JTextField("200", 5);
        JButton drawBut = new JButton("draw");
        Container c = getContentPane();
        JPanel pNorth = new JPanel();
        pNorth.add(tf);
        c.add(pNorth, "North");
        JPanel pSouth = new JPanel();
        pSouth.add(drawBut);
        c.add(pSouth, "South");
        c.add(p, "Center");
    }
}
```

```
import javax.swing.*;
import java.awt.*;

public class CircleGuiPanel extends JPanel {

    private int diam = 200;

    public CircleGuiPanel ( ) {
        setBackground(Color.white);
    }

    public void paintComponent (Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.magenta);
        g.fillOval(196-diam/2, 150-diam/2, diam, diam);
    }
}
```

```
public class Main {

    public static void main (String [ ] args) {
        CircleGuiFrame win = new CircleGuiFrame();
        win.setVisible(true);
    }
}
```

The assessment task

Modify the code for the classes `CircleGuiFrame` and `CircleGuiPanel` so that if a positive integer n , say, is displayed in the text field at the top of the window (e.g. by selecting the text field with the mouse and typing from the keyboard) and the 'draw' action button pressed, the circle is redrawn with diameter n but with the same centre.

Further modify the code for the classes `CircleGuiFrame` and `CircleGuiPanel` to add an additional action button 'toggle' to the bottom of the window. When this action button is pressed the diameter and position of the circle is unchanged but instead it toggles between solid and hollow.



To earn your 2 marks for Lab Assessment 8, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 12 (the week beginning 18th October)**.

Lab Assessment 9

Before attempting this assessment you will need to complete your study of Chapter 11 of *Java Genesis*.



PROBLEM: processing marks

Students' marks are stored in a text file called *marks.txt*. In this file each line records a student's name followed by a sequence of marks stored as integers separated by white space. Here is a typical example of what the file *marks.txt* might look like (this file can be found in the *genesis/projects/Lab Assessment/Processing Marks* folder):

```
Anne Other 15 19 15 12 18
Roger the Dodger 11 13 9 15
Eric the Red 19 19 20 15 20
Java Janet 20 20 20
```

Construct a class `ProcessingMarks` which reads the file *marks.txt* and creates a text file *results.txt* where each line consists of a student's name just as in the file *marks.txt*, but this name is followed by the sum of that student's marks. For example, if the class `ProcessingMarks` is run and the above example of the file *marks.txt* is in the same folder, the file *results.txt* would become:

```
Anne Other 79
Roger the Dodger 48
Eric the Red 93
Java Janet 60
```

□

To earn your 2 marks for Lab Assessment 9, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 12 (the week beginning 18th October)**.

Lab Assessment 10

Before attempting this assessment you will need to complete your study of Section 12.1 of *Java Genesis*.

PROBLEM: oval pattern applet



Revisit Problem 5 *creating patterns with ovals* of Chapter 9 of *Java Genesis*. (The code for the classes `OvalFrame` and `OvalPanel` can be found in the folder *genesis/projects/Lab Assessment/Oval Pattern Applet*). When the `Main` class is run a window opens displaying a pattern created by drawing various oval shapes.

Modify `OvalFrame` to create a class `OvalApplet` that extend the `JApplet` class. Also create an associated HTML file *OvalApplet.html* so that when this file is selected as the main HTML file and run in the applet viewer, the oval pattern previously displayed as a window application is now displayed as an applet.



To earn your 2 marks for Lab Assessment 10, when you have completed this problem demonstrate your solution to a tutor during an allocated lab session **before the end of teaching week 12 (the week beginning 18th October)**.