

Digital Skills Academy

FUNDAMENTALS OF PROGRAMMING

LAB 4 – PROBLEM SHEET



© Copyright Digital Skills Academy 2011-12. All rights reserved.



Template LAB solution powerpoint

Problem Definition

Design

Draw object diagram and/or pseudo-code

Implementation

Translate your design into code, do so in a number of steps adding additional functionality with each step

Test Definition and Test Recording

Code

Screens

Documentation

Usage and maintenance documentation

References

All lab work should have associated documentation in the following format

Use the powerpoint template provided later we will use Web tools

PROBLEM 4.1 – AREA OF A TRIANGLE

Similar to the App and Sphere classes shown on the next slides. Write an application to calculate the area of a Triangle, where the area calculation is performed in a Triangle class not in the App class

The area of a triangle is = half the base * the perpendicular height

Objectives

- To create the default App class with main() only instantiating the App class
- To create an instantiable class

App.java

```

1  /*****
2  * calculate the volume of a sphere given the radius
3  * using a sphere class
4  *
5  * Date: 01 Oct 2011
6  * @author Conor O'Reilly
7  *
8  *****/
9
10 import javabook.*;
11
12 class App
13 {
14
15     public static void main(String args[])
16     {
17         App thisProgram = new App();
18     }
19
20     public App()
21     {
22         //Declare variables
23         double aRadius;
24         double aVolume;
25
26         //Declare objects
27         MainWindow mWindow;
28         InputBox iBox;
29         OutputBox oBox;
30
31         Sphere aSphere;
32
33         //Create objects
34         mWindow = new MainWindow();
35         iBox = new InputBox(mWindow);
36         oBox = new OutputBox(mWindow);
37
38         aSphere = new Sphere();
39
40         //Use objects
41         mWindow.show();
42
43         //get Input
44         aRadius = iBox.getDouble("Please enter the radius of the sphere: ");
45         aSphere.setTheRadius(aRadius);
46
47         //debug code
48         System.out.println( "value input:" + aRadius);
49         System.out.println( "the radius value in the sphere object:" + aSphere.getTheRadius() );
50
51         //Process
52         aSphere.computeVolume();
53         aVolume = aSphere.getTheVolume();
54
55         //Output
56         oBox.show();
57         oBox.print( " The volume of a sphere with a radius of " + aRadius + " is " + aVolume );
58     }
59 }
60
61 }
62

```

Sphere Class

```
Sphere.java App.java
6  *****/
7
8  class Sphere
9  {
10     // DATA
11     //.....
12     //Private Constants
13     private final double PI = Math.PI;
14     private final double RATIO_4_OVER_3 = (float) 4 / 3 ;
15
16     //Private Variables
17     private double theRadius;
18     private double theVolume;
19
20     // CONSTRUCTORS
21     //.....
22     public Sphere()           // same name as the class and the file name
23     {
24         this.theRadius=0;
25         this.theVolume=0;
26     }
27
28     // METHODS - behaviours
29     //.....
30     public void computeVolume()
31     {
32         this.theVolume = RATIO_4_OVER_3 * PI * ( this.theRadius * this.theRadius * this.theRadius);
33
34         // using the Math library the formula would be written
35         // this.theVolume = RATIO_4_OVER_3 * Math.PI * Math.pow( this.theRadius, 3);
36     }
37
38     // METHODS - gets (accessors) and sets (mutators)
39     //.....
40     public void setTheRadius(double radius)
41     {
42         this.theRadius= radius;
43     }
44
45     public double getTheRadius()
46     {
47         return(this.theRadius);
48     }
49
50     // SHOULD NOT HAVE THIS METHOD, as theVolume is set by computeVolume()
51     // just an example of the use of private - so external classes cannot see this method
52     // and we wouldn't want it set any other way but by the computeVolume method.
53     // Even though it is private it would be better not to have this method at all.
54     private void setTheVolume(double volume)
55     {
56         this.theVolume = volume;
57     }
58
59
60     public double getTheVolume()
61     {
62         return(this.theVolume);
63     }
64
65 }
66
```

PROBLEM 4.2

Similar to the App and Sphere classes shown on in 4.1 design and write an App and Dice class. On execution the App class should create three Dice objects and toss each one. The results of the toss should be displayed in an output window

The dice class design is given on the next slide

Objectives

- To create the default App class with main() only instantiating the App class
- To create an instantiable class
- To create multiple objects of a class

The dice class

Dice.java

```
1  /*****
2  *
3  *  Dice
4  *
5  *  Date: 14 Oct 2011
6  *  @author Conor O Reilly
7  *
8  *  Maths.Random() returns a positive number between 0 and 1
9  *  to get a number in a range that you want e.g. 1 to 6 for a dice
10 *  you multiply the value produced by the random method by your upper bound e.g. 6
11 *  and add on the value of your lower bound e.g. 1 in this case
12 *
13 *****/
14
15 class Dice
16 {
17     // DATA
18     //.....
19     //Private Constants
20     final int NUMBER_OF_SIDES = 6;
21
22     //Private Variables
23     private int faceValue;
24
25
26     // CONSTRUCTORS
27     //.....
28     public Dice()
29     {
30         this.faceValue = 0;    //zero if not thrown
31     }
32
33     // METHODS - behaviours
34     //.....
35     public void throwDice()
36     {
37         this.faceValue = 1 + (int) (Math.random() * NUMBER_OF_SIDES);
38     }
39
40
41     // METHODS - gets (accessors) and sets (mutators)
42     //.....
43     public int getFaceValue()
44     {
45         return(this.faceValue);
46     }
47
48 }
49
```

PROBLEM 4.3

Based on the code on 4.2 ask the user if they want to throw one dice, two dice or three dice based on the answer roll the required number of dice and return the results.

Hint: Use IF based on the number of dice input

Objectives

- To create the default App class with main() only instantiating the App class
- To create an instantiable class
- To create multiple objects of a class only when required, so declare but no new is some cases
- Implementation of an if ... else if ... else statement

PROBLEM 4.4

Create a class that takes as input the test score a student obtained and outputs their grade.

Test Score	Grade
$90 \leq \text{score}$	A
$80 \leq \text{score} < 90$	B
$70 \leq \text{score} < 80$	C
$60 \leq \text{score} < 70$	D
$\text{score} < 60$	F

```
if (score >= 90)
    mBox.show("Your grade is A");
else if (score >= 80)
    mBox.show("Your grade is B");
else if (score >= 70)
    mBox.show("Your grade is C");
else if (score >= 60)
    mBox.show("Your grade is D");
else
    mBox.show("Your grade is F");
```

Objectives

- To create the default App class with main() only instantiating the App class
- To create an instantiable class
- To create multiple objects of a class only when required, so declare but no new is some cases
- Implementation of an if ... else if ... else statement

PROBLEM 4.5

Create a leap year class that takes as input the a year.

Outputting if the date is a leap year or not with the rules that were applied.

Leap year rules are:

Is the year evenly divisible by 4? If so, it is a leap year,
unless...

Is the year evenly divisible by 100? (for example, 1500?) If so, it is not
a leap year,
unless...

Is the year evenly divisible by 400? If so, it is a leap year.

Objectives

- To create the default App class with main() only instantiating the App class
- To create an instantiable class
- To create multiple objects of a class only when required, so declare but no new is some cases
- Implementation of an if ... else if ... else statement
- [Ref : http://support.microsoft.com/kb/214019/EN-US](http://support.microsoft.com/kb/214019/EN-US)