**COEN 160	OO Analysis, Design and Programming	Fall 2024**

Description: In this lab, you will start up Eclipse, create a project with packages, create classes, and fix errors. We will go through all of this together, except for fixing the errors.

Part I

Exercise 1

1. Enter: “setup eclipse”
2. Enter: “eclipse”
3. Eclipse will ask you to specify a workspace. A workspace is an actual physical directory on your machine. When a workspace is selected, Eclipse will load in any projects within that directory. You may also add and edit projects inside this directory as well. For now, simply just specify a directory (**coen160labs**, for example) where you intend to keep your project files.
4. If you see the welcome screen, simply click “workbench” to continue.
5. Go to **File->New->Java Project**. Fill a name for the project (**Lab1**, for example) deselect “Create module-info.java” and click “Finish”. Your project should appear in the Package Explorer. If you cannot click “Finish”, then click “Next” twice.
6. Within your project in the Package Explorer, you should see a “src” folder. Right click the source folder, and go to **New->Package**. A window should appear. Give the package a name **(lab1_1_1**). Click “Finish”.
7. Right click the package you just created within the Package Explorer. Go to **New->Class**. A window should appear. Give the class the name “Puppy” and make sure the Modifier is set to public. Click “Finish”. A java file should appear under the package within the Package Explorer.
8. If the java file isn’t open, double click the java file in the Package Explorer. Type the given code into the java file.
9. Fix the errors.
10. Edit the code, so that the age (in years) of the puppy is printed from main.
11. Edit the code, so that the age (in days) is printed from main.

```
package lab1_1_1;

public class Puppy {
      private String name;
      private Int age;

      public Puppyy(){
            that.name = "Name not given yet’;
            this.age = 1;
      }

      public Puppy(Strong name, int age){
            that.name = name;
            this.age = “age”;
      }

      public String getName() {
            return this.nome;
      }

      public static void main(String args) {
            Puppy myPuppy = new Puppyy("Fido",2);
            System.out.println("Puppy name: "+myPuppy.gotName());
      }
}
```

Exercise 2

- Create a new package lab1_1_2
- Create a new class called Cube
- Copy the following code into it.
- Identify the errors
- Fix and compile it.

```
Public class Cube
{
   public static void main()
   {
       double height = 3.0; \ inches
       double cube-volume = height * height * height
       System.out.println("Volume = " cube_volume);
    }
```

Exercise 3

In this exercise, you will learn to use static methods and get practice on using loops.

- Create a new package lab1_1_3
- Create a class called PatternMaker and copy the code given below
- Complete the class **PatternMaker** (given below) with two static methods:
    - **drawOneLine()**
    - **drawPattern()**
- Use the comments and complete the methods.
- Execute it.

```
class PatternMaker {
      public static void drawOneLine(char symbol, int noOfTimes, int
offset){
            // Write Java code to draw the symbol for the noOfTimes
            // after drawing a number of blankspaces (offset) .
            // For example, if the symbol is "*", noOfTimes is 10
            // and offset is 3,it should draw 3 blank spaces and
            // then draw 10 stars.

            //System.out.println(); // Print new line.

           // Write a for – loop to draw offset number of blank spaces.
            // for example, if offset is 3, it should draw 3 blank
            // spaces. Use System.out.print(" ") to print a single
            // blank space. Write a for – loop to draw the symbol,
            // the noOfTimes.

      }

      public static void drawPattern(){
            // This method should draw the following pattern:
            // 1. line, should have 4 blank spaces followed by 4 stars.
            // 2. line will have 8 blank spaces followed by 8 stars.
            // 3. line will have 12 blank spaces followed by 12 stars.
            //     ****
            //         ********
            //             ************
            // The method should call the method, drawOneLine()
            // with the correct number of values for parameters.
      }

      public static void main (String [] args){
      }
}
```

Part II

Exercise 4

- Create a package lab1_2_4.
- Copy the file, **ScannerDemo.java** underneath this folder.
- In this program, you will learn how to use a **Scanner** class in Java, to read keyboard input (and in later weeks, input from a file).
- The Scanner class is a class in the package, *java.util*, which allows the user to read values of various types.
- We will use this class, for now, to read numbers (integers and reals).
- Compile and run the program.
- You will be prompted to enter input. Enter an integer of your choice on a prompt for an integer.
- Enter an integer of your choice on a prompt for a float.
- Enter a real number with a decimal point and digits after decimal point of your choice on a prompt for a double.
- Did the output correctly show the values you have entered?
- Now, run the program again. Enter a floating point number (with a decimal and digits after decimal) of your choice on a prompt for an integer. Did the output show correctly?
- You will use the class **Scanner** to read in user input in Exercise5.
  Ref: http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

Exercise 5

In this exercise, you will use **modulo division (%) and a conditional statement** in Java to check if a given year is a leap year or not.

- Create a package lab1_2_5.
- Copy the code of **Exercise5.java**.
- Complete the method, **isItLeapYear()**.
- You will write the test code in the main function and test your method using the test data written in the comments.
- In part b) of this exercise, you will use the Scanner class to allow the user to input test-data of his/her choice.

Based on your output, answer the following questions.

a) Show (of the years given), the leap years.

b) Now, use the Scanner class you have used in the ScannerDemo.java to read the user input for years. In the Exercise5.java, in class Exercise5, write a static method called tester(). In this method, write code to create an instance of a Scanner to read in the user's input (as integers) for years. Check if the year input is a leap year or not.
Test your *tester()* by calling it from main() in Exercise5.java.

c) The method, *isItALeapYear()* is written as a class-level (static) method. Rewrite the method as an instance method. Comment out the testing code you have in tester () method and insert code to test your instance method. Use the same years to test your code.

Exercise 6

In this exercise, you will use the Java class String operations to complete the code segment given.
Refer to http://docs.oracle.com/javase/6/docs/api/java/lang/String.html, for String API.

- Create a package lab1_2_6.
- Copy the code in the Java file, **Exercise6.java**.
- You are required to complete the two methods:
    - **fullName()**
    - **palindrome()**
- You will write the test code in **Exercise6** and test your methods.
- **fullName():** Complete the method using the comments given. For a reference to String API, check the link given above.
- **palindrome():** A palindrome is a word, phrase or a sentence that reads the same when read backwards.
- The String class in Java does not have a method to reverse a string. There is another Java class called StringBuffer which has the method *reverse()*, that reverses a string.
- String objects in Java are immutable (cannot be changed), but **StringBuffer** objects are mutable.

Check the code given below to reverse a string using a StringBuffer object.

```
// original String
String strOriginal = "Hello World";

// create a StringBuffer object
StringBuffer tempStr = new StringBuffer(strOriginal);

// reverse the StringBuffer object
tempStr = tempStr.reverse();

// convert it back to a String, using toString()
String strReversed = tempStr.toString();

// print it
System.out.println("Reversed String : " + strReversed);
```

Points to note:

a) When you are checking for a palindrome, ignore case. That is, Madam is a palindrome if you ignore case. Otherwise, it is not.
b) When you are checking a sentence to see if it is a palindrome (“A man, a plan, a canal, Panama”, for example), you may have to consider just the text without commas and spaces (and ignore case as well).

Some of the String methods you may consider using (if and where applicable) are:

a) equals()
b) charAt()
c) substring()
d) toLowerCase()
e) toUpperCase()
f) length()

Please check the String API (link given above) and select the methods to use.

Random Numbers in Java

There are two ways to generate random numbers in Java, namely, using the methods in class Random or using the random() method in Math class.

- The Random class generates random integers, doubles, longs and so on, in various ranges.
- The static method `Math.`random generates *doubles* between 0 (inclusive) and 1 (exclusive).

Exercise 7

- Create a package lab1_2_7.
- In **Exercise7.java**, you will use the random() method in Math class.
- Complete the method, **genRandomNum()**, using the comments given.
- Test your code.