Java Programming Laboratory

Classes and Objects

Exercise 1: String class. This exercise focuses on the String class defined in the Java Standard Class Library. Create a controller Java class with a main method and undertake the tasks below.

1. Declare a variable called str to be a reference to a String object as shown below.

```
String str = new String("happy coding");
```

2. You use the dot-operator to invoke a method to operate on an object. Display the length of str using the length method of the String class as shown below.

```
System.out.println("length = "+str.length());
```

3. Again using the dot-operator use the toUpperCase() method of the String class to convert all the lower case letters to upper case as shown below.

```
System.out.println(str.toUpperCase());
```

Exercise 2: Maths class. This exercise focuses on the Maths class defined in the Java Standard Class Library . Create a controller Java class with a main method and undertake the tasks below.

1. The methods in the Math class are static methods and so you do not need to create objects to use them. Display the square root of 4 using the static Math.sqrt() method as shown below.

```
System.out.println("SQRT of  4 = "+Math.sqrt(4));
```

2. Calculate and display 10 to the power 2 using the Math.pow() method as shown below.

```
System.out.println("Power(10,2) = "+Math.pow(10, 2));
```

Exercise 3: Random class. This exercise focuses on the Random class defined in the java.util package. Create a controller Java class with a main method and undertake the tasks below.

1. Import the Random class as shown below.

```
import java.util.Random;
```

Put the import declaration at the top of your program. Note that you do not need to use an import statement for classes defined in the java.lang package (this includes String and Math).

2. Next, create a Random object as shown below

```
Random rand = new Random();
```

3. Normally the random numbers to be generated need to be from a certain range (say between 1 to 10 inclusively). The nextInt() method of the Random class takes an integer parameter to denote the upper limit for the range of numbers. However, the nextInt() method works from zero up to the upper limit. This means that nextInt(10) will only select a random number from 0 to 9 inclusively. Using a for-loop generate five random numbers from 0-9 as shown below.

```
for (int i=0; i < 5;i++)
 {
   System.out.print(rand.nextInt(10)+ " ");
 }
```

To generate a random number between 1 to 10 inclusively add one to the result, use

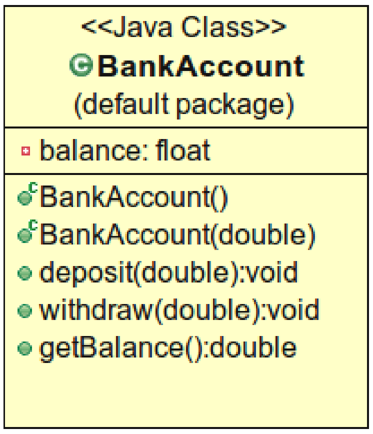
```
rand.nextInt(10)+1
```

4. The Random class generates random numbers in a deterministic way. The algorithm that produces the randomness is based on a number called a seed. You can use System.currentTimeMillis() as the seed to produce random results. Using a for-loop generate five random double values as shown below.

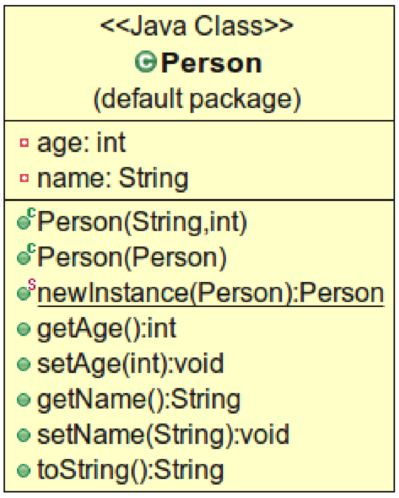
```
Random rand2 = new Random (System.currentTimeMillis());

  for (int i=0; i < 5;i++)
  {
        System.out.print(rand2.nextDouble()+ " ");
  }
```

The method nextDouble() of the Random class returns a random double value between 0.0 and 1.0

Exercise 4: BankAccount class Implement the bank account class defined by the UML diagram below and develop a controller class to test the class by creating an account object and print details to the console screen.



Exercise 5: Person class. Write a Person class with two fields, name and age, which has a copy constructor as shown in the UML diagram below. Write a controller class which creates two Person clones testing the copy constructor printing details to the console screen.



Exercise 6: A quadratic equation is any equation in the form $ax^2 + bx + c = 0$ where $a \neq 0$. Any quadratic equation can be solved with the formula $x = (-b \pm \sqrt{(b^2 - 4ac)})/2a$. Write a class called PolySolve which finds the roots of a quadratic equation and test your class using a Controller class with a main method.

Exercise 7: The formula for calculating the roots of a quadratic equation in exercise 6 becomes infinity when a =0. Modify your PolySolve class to make it more robust so that it can deal with cases where a=0 (i.e. solving 10x -8). Also add a method to the PolySolve class called *display* which print out the polynomial and roots and a method called *add* which will take a single argument of type PolySolve and return the sum of the two polynomials. Test your new methods with the Controller class.

Dr Alan Crispin
14-09-15