Lab3

March 20, 2025

1 Lab 3

Deadline: **Week 5** in your respective lab session

1.0.1 Name: Kiril Radev

1.0.2 Student ID: 241005831

1.1 Question 1 [1 mark]

Write class `Utils`, which will contain useful constants and methods. It should contain a `final` class variable `PI` and store a value `3.14`. The class `Utils` should also contain two class methods, `circlePerimeter` and `circleArea`.

`circlePerimeter` should take the circle's radius as an argument, calculate and return the circle's perimeter with such radius with exactly 1 decimal place rounded down.

The formula for the perimeter of a circle is:
$circle\ \ perimeter = 2\pi r$ where r is a radius.

`circleArea` should take the circle's radius as an argument, calculate and return the area of the circle with such radius with exactly 1 decimal place rounded down.

The formula for the area of a circle is:
$circle\ \ area = \pi r^2$ where r is a radius.

You are NOT allowed to use libraries or modify a String representation of the number to achieve this.

Lastly, write class `Main1` with the main method that asks the user to input the circle's radius and prints out the value of , the circle perimeter and the area with such radius. To achieve this, you must use a variable and methods defined within `Utils`.

Example run:

```
What is the radius of your circle? 26
The value of   is 3.14
The perimeter of your circle is 163.2 units
The area of your circle is 2122.6 square units
```

Write your answer below:

#1 Main1

[3]:
```
import java.util.Scanner;

public class Main1
{
        public static void main(String[] args)
        {
                Scanner scanner = new Scanner(System.in);
                Utils circle_calculator = new Utils();
                //1
                System.out.print("What is the radius of your circle? ");
                double r = Double.parseDouble(scanner.nextLine());
                //2
                System.out.println("The value of   is 3.14");
                //3
                double perimeter = circle_calculator.circlePerimeter(r);
                System.out.println("The perimeter of your circle is " +␣
 ↪perimeter + " units");
                //4
                double area = circle_calculator.circleArea(r);
                System.out.println("The area of your circle is " + area + "␣
 ↪square units");

        }//END main

}//END Mini1
```

#2 Utils

[4]:
```
public class Utils
{
        //constant global variable PI
        final double PI = 3.14;

        public double circlePerimeter(double r)
        {
                double perimeter = 2 * PI * r;
                perimeter = roundByOneDecimalPlace(perimeter);

                return perimeter;
        }//END circlePerimeter

        public double circleArea(double r)
        {
                double area = PI * r * r;
                area = roundByOneDecimalPlace(area);

                return area;
        }//END circleArea

        //basic tool method
        private double roundByOneDecimalPlace(double to_round)
        {
                to_round = (int)(to_round * 10) / 10.0;

                return to_round;
        }//END roundByOneDecimalPlace

}//END
```

Run your program:

[5]:
```
Main1.main(null)
```

```
What is the radius of your circle?

 24

The value of   is 3.14
The perimeter of your circle is 150.7 units
The area of your circle is 1808.6 square units
```

1.2 Question 2 [1 mark]

Write a class `Dictionary` which can store up to `10000` words and their definitions.

The class `Dictionary` has two instance methods `addEntry` and `findDefition`.

`addEntry` takes two Strings as arguments, a word and its definitions and adds them to the array with all the other entries.

`findDefinition` takes a String as an argument, a word, and checks whether it was entered in the dictionary. If found, it returns its definitions; otherwise, it returns `"Not Found."`.

Solutions that concatenate a word and its definition into 1 String to store them inside the array will not be accepted. We are looking for an Object-Oriented solution.

Hint: Write another class, `DictionaryEntry`.

Write your answer below:

#1 Dictionary

[8]:
```
public class Dictionary
{
        private final int MAX_SIZE = 10000;
        private DictionaryEntry[] list;
        private int new_index;

```

```java
        //constructor
        Dictionary()
        {
                this.list = new DictionaryEntry[MAX_SIZE];
                this.new_index = 0;
        }

        public void addEntry(String word, String definition)
        {
                DictionaryEntry new_entry = new␣
↪DictionaryEntry(word,definition);
                //check entries size
                if(new_index == MAX_SIZE)
                {
                        System.out.println("You have reached the maximum amount␣
↪of words in the dictionary");
                        return;
                }

                list[new_index] = new_entry;
                new_index++;

        }//END addEntry

        public String findDefinition(String word)
        {
                //linear search
                for(int i = 0; i < new_index; i++)
                {
                        if(list[i].getWord().equals(word))
                        {
                                return list[i].getDefinition();
                        }
                }

                return "Not found.";
        }//END findDefinition

}//END Dictionary
```

#2 DictionaryEntry

[9]:
```java
public class DictionaryEntry
{
        private String word;
        private String definition;

```

```
    //constructor
        DictionaryEntry(String new_word, String new_definition)
        {
                this.word = new_word;
                this.definition = new_definition;
        }

    //accessor methods
        public String getWord()
        {
                return word;
        }

        public String getDefinition()
        {
                return definition;
        }

}//END DictionaryEntry
```

#3 Main2

[10]:
```
public class Main2
{
    public static void main(String[] args)
    {
        Dictionary englishLanguage = new Dictionary();

        englishLanguage.addEntry("dog", "A loyal mammal, domesticated for␣
 ↪companionship, belonging to the Canidae family.");
        englishLanguage.addEntry("cat", "A cat is a domesticated feline mammal,␣
 ↪valued for companionship, belonging to the Felidae family.");

        System.out.println(englishLanguage.findDefinition("dog"));
        System.out.println(englishLanguage.findDefinition("cat"));
        System.out.println(englishLanguage.findDefinition("octopus"));
    }//END main

}//END Main2
```

Run your program:

[11]:
```
Main2.main(null);
```

```
A loyal mammal, domesticated for companionship, belonging to the Canidae family.
A cat is a domesticated feline mammal, valued for companionship, belonging to
the Felidae family.
Not found.
```

1.3 Question 3 [1 mark]

Paste your code from Question 2 below and modify it (start by renaming each class) to implement the Singleton Design Pattern so that you can create only one instance of a class `Dictionary`. If you attempt to create a second instance of a `Dictionary`, print out an error message `"This class is a singleton!"` and return `null`.

Singleton is a design pattern in software engineering. It is used to ensure there is only one instance of a particular class. It prevents us from accidentally creating more instances than we want to have.

For example, if we are developing a system to manage books in the library, we would only want one instance to keep track of all the books. Accidentally creating multiple instances storing data about all the books would (unintentionally) lead to numerous inconsistencies.

#1 Dictionary2

[27]:
```
//Singleton Pattern
public class Dictionary2
{
        //1.Private Static Instance of the class.
        private static Dictionary2 dictionary;
        //-- other objects used within the class
        private final int MAX_SIZE = 10000;
        private DictionaryEntry[] list;
        private int new_index;

        //2. Private constructor to avoid instantiation.
        private Dictionary2()
        {
                this.list = new DictionaryEntry[MAX_SIZE];
                this.new_index = 0;
        }

        //3. Public static method to provide access to the singleton(object)
        //   by getting the instance variable and checking if it refers to␣
 ↪nothing.
        //   If it refers to something, then we are trying to initialise a new␣
 ↪object,
        //   meaning we disobey the rules of a singleton, that is why we have␣
 ↪the else statement
        //   that tells us if we are making the mistake of creating a mew␣
 ↪Dictionary2.
        public static Dictionary2 createDictionary2()
        {
                if(dictionary == null)
                {
                        dictionary = new Dictionary2();
                        return dictionary;
                }
                else
                {
                        System.out.println("This class is a singleton!");
                        return null;
                }
        }


        //-- individual public methods of the class
        public void addEntry(String word, String definition)
        {
                DictionaryEntry new_entry = new␣
 ↪DictionaryEntry(word,definition);
                //check entries size
                if(new_index == MAX_SIZE)
                {
                        System.out.println("You have reached the maximum amount␣
 ↪of words in the dictionary");
                        return;
                }

                list[new_index] = new_entry;
                new_index++;

        }//END addEntry

        public String findDefinition(String word)
        {
                //linear search
                for(int i = 0; i < new_index; i++)
                {
                        if(list[i].getWord().equals(word))
                        {
                                return list[i].getDefinition();
                        }
                }

                return "Not found.";
        }//END findDefinition

}//END Dictionary
```

#2 DictionaryEntry

[22]:
```
public class DictionaryEntry
{
        private String word;
        private String definition;

        DictionaryEntry(String new_word, String new_definition)
        {
                this.word = new_word;
                this.definition = new_definition;
        }

        public String getWord()
        {
                return word;
        }

        public String getDefinition()
        {
                return definition;
        }

}//END DictionaryEntry
```

#3 Main3

[28]:
```
class Main3 {
    public static void main(String[] args) {
        Dictionary2 englishLanguage = Dictionary2.createDictionary2();
        Dictionary2 welshLanguage = Dictionary2.createDictionary2();

        englishLanguage.addEntry("dog", "A loyal mammal, domesticated for␣
 ↪companionship, belonging to the Canidae family.");
        englishLanguage.addEntry("cat", "A cat is a domesticated feline mammal,␣
 ↪valued for companionship, belonging to the Felidae family.");

        System.out.println(englishLanguage.findDefinition("dog"));
        System.out.println(englishLanguage.findDefinition("cat"));
        System.out.println(englishLanguage.findDefinition("octopus"));

        System.out.println(welshLanguage); // null
    }
}
```

Run your program:

[29]:
```
Main3.main(null);
```

```
This class is a singleton!
```

```
A loyal mammal, domesticated for companionship, belonging to the Canidae family.
A cat is a domesticated feline mammal, valued for companionship, belonging to
the Felidae family.
Not found.
null
```

1.4 Question 4 [1 mark]

Write class `Animal`, which stores the animal's name and age. It also contains a constructor and a method `displayInfo` that prints out information about the animal. The class `Animal` has 2 sub-classes `Dog` and `Cat`.

`Dog` has an instance variable `breed`, which is initialised through its constructor. It also has a method `bark` that prints out a bark sound and the dog's breed.

`Cat` has an instance variable `hasClaws` which is initialised through its constructor. It also has a method `meow` that prints out a meow sound and whether a cat has claws.

Lastly, define class `Main4` to test your code.

Write your answer below:

#1 Animal

[30]:
```
public class Animal
{
        private String name;
        private int age;

        //constructor
        Animal(String new_name, int new_age)
        {
                this.name = new_name;
                this.age = new_age;
        }

        public void displayInfo()
        {
                System.out.println("Name:  " + name);
                System.out.println("Age :  " + age);
        }//END displayInfo

}//END Animal
```

#2 Dog

[31]:
```
public class Dog extends Animal
{
        private String breed;
        //constructor
        Dog(String new_name, int new_age, String new_breed)
        {
                super(new_name, new_age);  //extension
                this.breed = new_breed;
        }

        public void bark()
        {
                System.out.println("'bark' " + breed);
        }//END bark

}//END Dog>>Animal
```

#3 Cat

[32]:
```
public class Cat extends Animal
{
        private boolean hasClaws;

        Cat(String new_name, int new_age, boolean new_claws)
        {
                super(new_name, new_age);  //extension
                this.hasClaws = new_claws;
        }

        public void meow()
        {
                if(hasClaws)
                {
                        System.out.println("meow");
                }
        }//END meow

}//END Animal>>Cat
```

#4 Main4

[34]:
```
class Main4 {
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy", 3, "Golden Retriever");
        myDog.displayInfo();
        myDog.bark();

        System.out.println();

        Cat myCat = new Cat("Whiskers", 2, true);
        myCat.displayInfo();
        myCat.meow();
    }//END main

}//END Main4
```

Run your program:

[36]:
```
Main4.main(null);
```

```
Name: Buddy
Age : 3
'bark' Golden Retriever

Name: Whiskers
Age : 2
meow
```

1.5 Question 5 [1 mark]

You have three identical bags, each containing two marbles.

Bag A contains 2 white marbles, bag B contains 2 black marbles, and Bag C contains 1 white and 1 black marble. You pick a bag at random and draw out 1 marble. If the marble is white, what is the probability that the other marble in the same bag is also white?

With three results:

- First marble was white n times
- Second marble was white n times
- Probability that 2nd marble was white when 1st marble was white: $n2$ / $n1$

In order to solve this puzzle you decide to implement a small Java program that simulates this experiment for 10k times. Write this program below.

Remember to `import java.util.Random`. Then you can do the following:

`Random random = new Random();`

`int bagIndex = random.nextInt(3);`

Selects a bag at random.

`int marbleIndex1 = random.nextInt(2);`

Picks a marble from the bag.

Write your answer below:

[12]:
```
import java.util.Random;
```

```
public class BagsOfMarbles
{
    public static void main(String[] args)
    {
            int[] results = {0,0};
            int experiment_num = 10000;

            for(int i = 0; i < experiment_num; i++)
            {
                    int[] temporary = bagsOfMarbles();
                    results[0] += temporary[0]; // P(B)
                    results[1] += temporary[1]; // P(A/\B)
            }

            //P(A|B) = P(A/\B) / P(B)
            int final_result = (int)(results[1] * 100.0/results[0]);

            System.out.println("First marble was white " + results[0] + " times.
 ↪");
            System.out.println("Second marble was white " + results[1] + "␣
 ↪times.");
            System.out.println("Probability that 2nd marble was white when 1st␣
 ↪marble was white: " + final_result + "%");
    }

    public static int[] bagsOfMarbles()
    {
            String[][] bags = {{"W","W"},  // bag A
                                   {"B","B"},  // bag B
                                   {"W","B"}}; // bag C

            int n1 = 0;  // first draw
            int n2 = 0;  // second draw

            Random random = new Random();
            int bagIndex = random.nextInt(3);       // pick random bag
            int marbleIndex1 = random.nextInt(2);  // take random marble

            if(bags[bagIndex][marbleIndex1].equals("W"))                  //␣
 ↪check 1st marble
            {
                    n1++;                                                     /
 ↪/ state the marble1 is white
                    if(bags[bagIndex][marbleIndex2(marbleIndex1)].equals("W")) /
 ↪/ check leftover marble
                    {
                            n2++;
↪   // state the marble2 is white
                    }
            }

            int[] results = {n1,n2};
            return results;
    }

    public static int marbleIndex2(int i1)
    {
            //if i1 2nt element
            if(i1 == 1)
                    return 0;
            //base case i1 1st element
            return 1;
    }
}
```

Run your program:

[13]:
```
BagsOfMarbles.main(null);
```

```
First marble was white 4984 times.
Second marble was white 3285 times.
Probability that 2nd marble was white when 1st marble was white: 65%
```

[ ]:
```

```