Lab4

March 6, 2024

1 Lab 4

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

1.0.1 Name:Hansel Natividade Rodrigues

1.0.2 Student ID: 230603866

You should only use things we learned up to this week (week 4), i.e. ArrayLists, HashMaps, Hashtables, etc., are not allowed. If you are unfamiliar with them, you are not expected to at this stage. Some of them will be introduced in future weeks.

1.1 Question 1 [1 mark]

Write class `Animal`.

Class `Animal` has two subclasses, `Dog` and `Cat`.

Override the `toString()` method in class `Animal` and its subclasses. It should now return the name of the class e.g. `"Animal"`.

Write a method `populateArray`, which takes an integer as an argument specifying the size of the array, creates a new array of specified size, populates it randomly with instances of `Dog` and `Cat` and returns it. Both classes should have an equal chance of being instantiated.

Lastly write some code to test the method you defined.

Write your answer below:

[2]:
```
import java.util.Random;

class Animal
{
    @Override
    public String toString()
    {
        return "Animal";
    }
}
```

[3]:
```
class Dog extends Animal
{
    @Override
    public String toString()
    {
        return "Dog";
    }
}
```

[4]:
```
class Cat extends Animal
{
    @Override
    public String toString()
    {
        return "Cat";
    }
}
```

[5]:
```
public class AnimalPopulator
{
    public static Animal[] populateArray(int size)
    {
        Animal[] animals = new Animal[size];
        Random random = new Random();
        for (int i = 0; i < size; i++)
        {
            if (random.nextBoolean())
            {
                animals[i] = new Dog();
            }
            else
            {
                animals[i] = new Cat();
            }
        }
        return animals;
    }

    public static void main(String[] args)
    {
        Animal[] animals = populateArray(12);
        for (Animal animal : animals)
        {
            System.out.println(animal);
        }
    }
}
```

Run your program:

[6]:
```
AnimalPopulator.main(null);
```

```
Dog
Cat
Cat
Cat
Cat
Cat
Cat
Dog
Cat
Cat
Dog
Cat
```

1.2 Question 2 [1 mark]

Write a method `separateArray` which separates an array of type `Animal` into two arrays and returns both inside another array (2D array). The first array should be of type `Dog` and contain just instances of class `Dog`, and the second should be of type `Cat` with just instances of class `Cat`.

Remember to test your code!

Write your answer below:

[7]:
```
public class AnimalSeparator
{
    public static Animal[][] separateArray(Animal[] animals)
    {
        int dogCount = 0;
        int catCount = 0;
        for (Animal animal : animals)
        {
            if (animal instanceof Dog)
            {
                dogCount++;
            }
            else if (animal instanceof Cat)
            {
                catCount++;
            }
        }
        Dog[] dogs = new Dog[dogCount];
        Cat[] cats = new Cat[catCount];
        int dogIndex = 0;
        int catIndex = 0;
```

```
        for (Animal animal : animals)
        {
            if (animal instanceof Dog)
            {
                dogs[dogIndex++] = (Dog) animal;
            }
            else if (animal instanceof Cat)
            {
                cats[catIndex++] = (Cat) animal;
            }
        }
        Animal[][] separatedArrays = new Animal[2][];
        separatedArrays[0] = dogs;
        separatedArrays[1] = cats;
        return separatedArrays;
    }

    public static void main(String[] args)
    {
        Animal[] animals =
        {
                new Dog(),
                new Cat(),
                new Dog(),
                new Cat(),
                new Dog(),
                new Cat(),
                new Cat(),
                new Cat(),
                new Dog()
        };
        Animal[][] separatedArrays = separateArray(animals);
        System.out.println("Dogs:");
        for (Animal dog : separatedArrays[0])
        {
            System.out.println(dog);
        }

        System.out.println("\nCats:");
        for (Animal cat : separatedArrays[1])
        {
            System.out.println(cat);
        }
    }
}
```

Run your program:

[8]:
```
AnimalSeparator.main(null);
```

```
Dogs:
Dog
Dog
Dog
Dog

Cats:
Cat
Cat
Cat
Cat
Cat
```

1.3 Question 3 [1 mark]

Write a method `countAnimalInstances` which takes an array of type `Animal` and returns an array with counts for each unique class.

For example `{new Cat(), new Dog(), new Cat()}` should return `{2, 1}`. This is because the instance of class `Cat` occurred `2` times, and the instance of class `Dog` occurred `1` time. The order of the counts should be based on the order in which instances occurred first in the array. Here, the cat occurred first; thus, it is placed at index `0`, whereas the dog occurred second; thus, it is placed at index `1`.

Another example: `{new Cat(), new Pig(), new Cat(), new Cat(), new Dog(), new Cow(), new Dog()}` should return `{3, 1, 2, 1}`.

You should assume the method must work for any number of subclasses of Animal, i.e. if we create subclasses `Pig` and `Cow`, the method should still work.

Write your answer below:

[11]:
```
class Pig extends Animal
{
    @Override
    public String toString()
    {
        return "Pig";
    }
}
```

[12]:
```
class Cow extends Animal
{
    @Override
    public String toString()
    {
```

```
        return "Cow";
    }
}
```

[9]:
```
public class AnimalCounter
{
    public static int[] countAnimalInstances(Animal[] animals)
    {

        int uniqueClasses = countUniqueClasses(animals);
        int[] counts = new int[uniqueClasses];
        int currentIndex = 0;
        for (int i = 0; i < animals.length; i++)
        {
            int count = 1;
            if (isNotCounted(animals, i))
            {
                for (int j = i + 1; j < animals.length; j++)
                {
                    if (animals[i].getClass() == animals[j].getClass())
                    {
                        count++;
                    }
                }
                counts[currentIndex++] = count;
            }
        }

        return counts;
    }

    private static int countUniqueClasses(Animal[] animals)
    {
        int count = 0;
        for (int i = 0; i < animals.length; i++)
        {
            if (isNotCounted(animals, i))
            {
                count++;
            }
        }
        return count;
    }

    private static boolean isNotCounted(Animal[] animals, int currentIndex)
    {
        for (int i = 0; i < currentIndex; i++)
```

```
        {
            if (animals[currentIndex].getClass() == animals[i].getClass())
            {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args)
    {
        Animal[] animals1 = {new Cat(), new Dog(), new Cat()};
        Animal[] animals2 = {new Cat(), new Pig(), new Cat(), new Cat(), new␣
 ↪Dog(), new Cow(), new Dog()};
        int[] counts1 = countAnimalInstances(animals1);
        int[] counts2 = countAnimalInstances(animals2);
        System.out.println("Counts for animals1: ");
        for (int count : counts1) {
            System.out.print(count + " ");
        }
        System.out.println();
        System.out.println("Counts for animals2: ");
        for (int count : counts2) {
            System.out.print(count + " ");
        }

    }
}
```

Run your program:

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

```
Counts for animals1:
2 1
Counts for animals2:
3 1 2 1
```

1.4 Question 4 [1 mark]

Write class `Person`, which has a private instance variable `name`, a constructor to initialise the `name` and a method `printInfo`, which prints out the person's name.

Write another class, `Student`, a subclass of a `Person`. `Student` has instance variable `SID`, constructor to initialise the `SID` and overrides method `printInfo`. It should print out the `name` and `SID`.

You are expected to use the keyword **super** in your answer.

Test your code!

Write your answer below:

[14]:
```java
public class Person
{

    private final String name;

    public Person(String name)
    {
        this.name = name;
    }

    public void printInfo()
    {
        System.out.println("Name: " + name);
    }
}
```

[15]:
```java
public class Student extends Person
{

    private final String SID;

    public Student(String name, String SID)
    {
        super(name);
        this.SID = SID;
    }

    @Override
    public void printInfo()
    {
        super.printInfo();
        System.out.println("SID: " + SID);
    }
    public static void main(String[] args)
    {
        Student student1 = new Student("Michael", "12823");
        Student student2 = new Student("John", "12902");
        Student student3 = new Student("Jack", "00112");
        student1.printInfo();
        student2.printInfo();
        student3.printInfo();
    }
```

```
}
```

Run your program:

[16]:
```
Student.main(null);
```

```
Name: Michael
SID: 12823
Name: John
SID: 12902
Name: Jack
SID: 00112
```

1.5 Question 5 [1 mark]

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where $0.1(6)$ means $0.166666\ldots$, and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

Tip: Make it work for $d < 10$ first.

Write your answer below:

[73]:
```
public class RecurringCycleFinder
{
    public static void main(String[] args)
    {
        int maxLength = 0;
        int denominatorWithMaxLength = 0;


        for (int d = 2; d <1000; d++)
        {
            int[] remainders = new int[d];
```

```
            int remainderValue = 1;
            int currentPosition = 0;
            while (remainders[remainderValue] == 0 && remainderValue != 0)
            {
                remainders[remainderValue] = currentPosition;
                remainderValue = (remainderValue * 10) % d;
                currentPosition++;
            }
            int cycleLength = currentPosition - remainders[remainderValue];
            if (cycleLength > maxLength)
            {
                maxLength = cycleLength;
                denominatorWithMaxLength = d;
            }
        }
        System.out.println("Value of d/1000 with the longest recurring cycle: "␣
 ↪+ denominatorWithMaxLength);
    }
}
```

Run your program:

[74]:
```
RecurringCycleFinder.main(null);
```

```
Value of d/1000 with the longest recurring cycle: 983
```

[ ]:

[ ]:

[ ]: