5COSC019W - Tutorial 3 Exercises

1 Implementing Constructors

The `VendingMachine` class implementing in the tutorial exercises of last week does not have any constructors. Instances of a class with no constructor are always constructed with all instance variables set to zero (or `null` if they are object references). It is always a good idea to provide an explicit constructor.

In this exercise, you should provide two constructors for the VendingMachine class:

- a default constructor that initialises the vending machine with 10 soda cans
- a constructor `VendingMachine(int cans)` that initialises the vending machine with the given number of cans

Both constructors should initialise the token count to 0.

What is the code for your constructors? Compile and run a test for your code.

2 Implementing a Class

Implement a class `Employee`. An employee has a a name and a salary (a `double`). The class should have a default constructor (i.e. one without any arguments), a constructor with two parameters (`name` and `salary`), and methods to return the name and the salary.

Write a program to test your class.

3 Designing and Implementing a Class

1. Implement a `VotingMachine` class that can be used for a simple election. It should have methods to clear the machine state, to vote for a Labour, to vote for a Conservative, and to get the number of votes for each of the two parties.
2. Assuming that your machine is biased towards one of the two parties, modify the `vote` method(s) to add a random number to the number of votes for that party.

   Hint: The `random()` method of the library class `Math`, can be used for the generation of a random number.

4 Objects are Copied by Reference

This exercise illustrates that objects are copied by reference, i.e. whether they are passed to a method, returned from a method or assigning an object's reference to another reference.

1. Guess what will be the output of the following program without running it:

```
class Cat {
    private String colour;

    public Cat(String colour1) {
        colour = colour1;
    }

    public String getColour() {
        return colour;
    }

    public void setColour(String colour1) {
        colour = colour1;
    }

    // creates a new Cat object and returns it to the caller
    public Cat create() {
        Cat c = new Cat("Brown");

        System.out.println("\nInside Cat.create(), address of created " +
                           " cat object c is: " +  c);

        return c;
    }
}


class Mutator {
    void mutate(Cat cat) {
        cat.setColour("Pink");
    }
}

public class ReferenceCopyExample {
    public static void main(String[] args) {
        Cat c1 = new Cat("Gray");

        Mutator mutator = new Mutator();
        mutator.mutate(c1);

        System.out.println("\nAfter mutate() is called, colour of cat c1 is: " +
                           c1.getColour());
```

```
        Cat c2 = new Cat("Black");  // create a new cat object

        System.out.println("After creation of cat c2");
        System.out.println("Memory address of object c1 is: " + c1);
        System.out.println("Memory address of object c2 is: " + c2 + "\n\n");

        c2 = c1;  // copy reference of c1 to c2

        System.out.println("After assignment c2 = c1;");
        System.out.println("Memory address of object c1 is: " + c1);
        System.out.println("Memory address of object c2 is: " + c2);


        c2.setColour("Yellow");  // set the colour of c2 to be yellow

        System.out.println("\n\nAfter c2.setColour(\"Yellow\")");
        System.out.println("Colour of c1 is: " + c1.getColour());
        System.out.println("Colour of c2 is: " + c2.getColour());

        Cat c3 = c1.create();
        System.out.println("Address of c3 is: " + c3);
    }
}
```

2. Verify your guess by running the program.

5 Object Comparison

1. Guess what will be the output of the following program without running it.

```
class MyInteger {
    private int i;

    // constructor
    public MyInteger(int m) {
        i = m;
    }
}

public class ObjectComparisonTest {
    public static void main(String[] args) {
        String g1 = "cat";
        String g2 = "cat";

        if (g1 == g2)
            System.out.println("g1 == g2 evaluates: true");
        else
            System.out.println("g1 == g2 evaluates: false");
```

```
        if (g1.equals(g2))
            System.out.println("g1.equals(g2) evaluates: true");
        else
            System.out.println("g1.equals(g2) evaluates: false");

        System.out.println();  // print an extra newline

        String h1 = new String("cat");
        String h2 = new String("cat");

        if (h1 == h2)
            System.out.println("h1 == h2 evaluates: true");
        else
            System.out.println("h1 == h2 evaluates: false");

        if (h1.equals(h2))
            System.out.println("h1.equals(h2) evaluates: true");
        else
            System.out.println("h1.equals(h2) evaluates: false");

        System.out.println();  // print an extra newline

        if (g1 == h1)
            System.out.println("g1 == h1 evaluates: true");
        else
            System.out.println("g1 == h1 evaluates: false");

        if (g1.equals(h1))
            System.out.println("g1.equals(h1) evaluates: true");
        else
            System.out.println("g1.equals(h1) evaluates: false");

        System.out.println();  // print an extra newline

        MyInteger m1 = new MyInteger(5);
        MyInteger m2 = new MyInteger(5);

        if (m1 == m2)
            System.out.println("m1 == m2 evaluates: true");
        else
            System.out.println("m1 == m2 evaluates: false");

        if (m1.equals(m2))
            System.out.println("m1.equals(m2) evaluates: true");
        else
            System.out.println("m1.equals(m2) evaluates: false");

    }
```

```
}
```

2. Verify your guess by running the program. If the output does not match your guess, justify the output that was produced.

6 The null keyword

The value `null` indicate that an object variable does not point to an object.

Guess what will happen when the following program is run. Verify your guess by running the program. Make sure you understand what happens.

```
public class NullObjectsTest {
    public static void main(String[] args) {
        String s = null;
        s.toUpperCase();
    }
}
```

7 Overloading Methods

The following class simulates a `Car`:

```
public class Car {
    private String licensePlate;
    private double speed;        // kilometers per hour
    private double maxSpeed;     // kilometers per hour

    // constructors
    public Car(String licensePlate1, double maxSpeed1) {
        this.licensePlate = licensePlate1;
        this.speed  = 0.0;
        if (maxSpeed1 >= 0.0) {
            maxSpeed = maxSpeed1;
        }
        else {
            maxSpeed = 0.0;
        }
    }

    /*******************************************/
    /* Code for overloaded constructor goes here */
    /*******************************************/
}
```

1. Implement an overloaded version of the constructor, which accepts an additional argument `double speed1`.

The value of the instance field `speed` should be set to the value of the passed argument `speed1`, according to the following constraints:

- If the value of `speed1` is negative, `speed` should be set to 0.
- If the value of `speed1` is greater than `maxSpeed1`, then `speed` should be set to `maxSpeed1`.

2. Implement a `print` method in the class, which prints the current speed, the maximum speed and the license plate of a `Car` object.

3. Implement a test class which tests the functionality of the `Car` class, by creating objects of the class using either of the constructors and calls method `print` on the created objects.

8 Challenge: Designing and Implementing Classes - The US Postal Service

Implement appropriate classes to solve the following problem.

For faster sorting of letters, the U.S. Postal Service encourages companies that send large volumes of mail to use a bar code denoting the ZIP code (see Figure 1 - left). The encoding scheme for a five-digit ZIP code is shown in Figure 1 - right. There are full-height frame bars on each side. The five encoded digits are followed by a check digit, which is com puted as follows: Add up all digits, and choose the check digit to make the sum a multiple of 10. For example, the sum of the digits in the ZIP code 95014 is 19, so the check digit is 1 to make the sum equal to 20. Each digit of the ZIP code, and the check digit, is encoded according to the following table, where 0 denotes a half bar and 1 a full bar.

```
Digit    Weight
        7 4 2 1 0
----------------
1       0 0 0 1 1
2       0 0 1 0 1
3       0 0 1 1 0
4       0 1 0 0 1
5       0 1 0 1 0
6       0 1 1 0 0
7       1 0 0 0 1
8       1 0 0 1 0
9       1 0 1 0 0
0       1 1 0 0 0
```

Note that they represent all combinations of two full and three half bars. The digit can be computed easily from the bar code using the column weights 7, 4, 2, 1, 0. For example, 01100 is:

```
 0x7 + 1x4+ 1x2+ 0x1+ 0x0 = 6
```

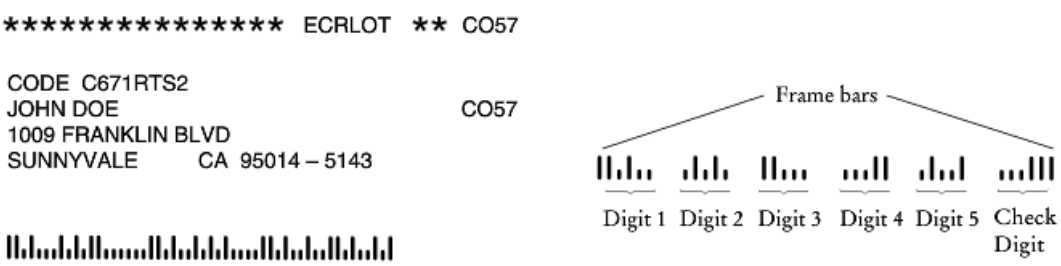

Figure 1: Left: A postal bar code. Zip code:95014. Right: Encoding for 5-digit bar codes

The only exception is 0, which would yield 11 according to the weight formula. Write a program that asks the user for a ZIP code and prints the bar code. Use : for half bars, | for full bars. For example, 95014 becomes:

|:|:::|:|:||:::::::||:|::|:::||

Your program should also be able to carry out the opposite conversion: Translate bars into their ZIP code, reporting any errors in the input format or a mismatch of the digits.