21/06034 - Samuel Wakoli
Task 3 Part 2

1. With relevant examples, explain the following concepts as used in Java programming.

a. Mutable classes.

Explain what is meant by mutable class

A mutable class is one that can change its internal state after it is created.

Write a program that implements the concept of mutable class

```
public class Main {
    private String myText;

    Main(String myText) {
        this.myText = myText;
    }

    public void setText(String myText) {
        this.myText = myText;
    }

    public String getText() {
        return myText;
    }

    //Main function
    public static void main(String[] args) {
    Main object = new Main("Dart is 2X faster than JavaScript and Java.");
    System.out.println(object.getText());
    // update the name using the setName method.
    object.setText("Flutter is the best tool for software development.");

    System.out.println(object.getText());
    }
}
```

b. Immutable classes.

Explain what is meant by immutable class

An immutable class is one that can not change its internal state after it is created.

Write a program that implements the concept of immutable class

```
public class Main {
    private final String text;
    Main(final String text) {
        this.text = text;
    }

    public final String getText() {
        return text;
    }

    //main method
    public static void main(String[] args) {
    Main obj = new Main("Kotlin is operable with Java.");
        System.out.println(obj.getText());
    }
}
```

c. Explain the situations where mutable classes are more preferable than immutable classes when writing a Java program.

- Immutable classes make it easier to parallelize your program as there are no conflicts among objects.
- The internal state of your program will be consistent even if you have exceptions.
- References to immutable objects can be cached as they are not going to change.
- Immutable classes are thread-safe so you will not have any synchronization issues.
- Immutable classes are good Map keys and Set elements, since these typically do not change once created.
- Immutable classes it easier to write, use and reason about the code (class invariant is established once and then unchanged)

2.

a) Explain what a String buffer class is as used in Java, the syntax of creating an object of StringBuffer class and Explain the methods in the StringBuffer class.

- String buffer is a thread-safe, mutable sequence of characters.
- Syntax for creating an object of StringBuffer class:
  ```
  StringBuffer objectName = new StringBuffer(String);
  ```
- Methods in the StringBuffer class:
  - length() - returns the length of the string i.e. total number of characters.
  - reverse() - returns the string in reversed order.

- append(argument) - used to append the specified string with another string. It can also be overloaded to support other data types.
- capacity() - returns the current capacity.

b) Write the output of the following program.

```
class Myoutput
{
    public static void main(String args[])
    {
        String ast = "hello i love java";
        System.out.println(ast.indexOf('e')+" "+ast.indexOf('ast')+" "+ast.lastIndexOf('l')+" "+ast .lastIndexOf('v'));
    }
}
```

Output:

The program does not execute. Hence we have no output due to errors within the code.

c) Explain your answer in (2b) above.

ast.indexOf('ast') caused a semantic error. indexOf() does not take a String argument hence resulting in an error.

d) With explanation, write the output of the following program.

```
class Myoutput

1.  {
2.      public static void main(String args[])
3.      {
4.          StringBuffer bfobj = new StringBuffer("Jambo");
5.          StringBuffer bfobj1 = new StringBuffer(" Kenya");
6.          c.append(bfobj1);
7.          System.out.println(bfobj);
8.      }
9.  }
```

The program does not execute due to another semantic error.
As from this line 6. `c.append(bfobj1);` There is no variable declaration of “c”.

e) With explanation, write the output of the following program.

```
class Myoutput
{
    public static void main(String args[])
    {
        StringBuffer str1 = new StringBuffer("Jambo");
        StringBuffer str2 = str1.reverse();
        System.out.println(str2);
    }
}
```

Output:

```
obmaJ
```

The output is “obmaj” because the original String “Jambo” has been reversed by the reverse() function.

f) With explanation, write the output of the following program.

```
class Myoutput
{
    class output
    {
      public static void main(String args[])
      {
         char c[]={'A', '1', 'b' ,' ' ,'a' , '0'};
         for (int i = 0; i < 5; ++i)
         {
                i++;
                  if(Character.isDigit(c[i]))
                      System.out.println(c[i]+" is a digit");
                  if(Character.isWhitespace(c[i]))
                      System.out.println(c[i]+" is a Whitespace character");
                  if(Character.isUpperCase(c[i]))
                      System.out.println(c[i]+" is an Upper case Letter");
                  if(Character.isLowerCase(c[i]))
                      System.out.println(c[i]+" is a lower case Letter");
                  i++;
          }
       }
}
```

Output:

```
1 is a digit
a is a lower case Letter
```

At the first loop, we check if the second value is a digit, a whitespace, an uppercase or lowercase. Since it is “1”, then it is a digit, and we print to the console.
We then skip the third value, and check the forth value if it is a digit, a whitespace, an uppercase or lowercase. Since the forth value is “a”, then it is a lowercase, and we print to the console.
We are skipping the first value, the third value and the fifth value because we are incrementing “i” two times in the loop.
We skipped the first value because “i” was initially having “0”