

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 Example {
    private String str;
    Example(String str) {
        this.str = str;
    }
    public String getName() {
        return str;
    }
    public void setName(String coursename) {
        this.str = coursename;
    }
    public static void main(String[] args) {
        Example obj = new Example("Diploma in IT");
        System.out.println(obj.getName());
        // Here, we can update the name using the setName method.
        obj.setName("Java Programming");
        System.out.println(obj.getName());
    }
}
```

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 Example {
    private final String str;
    Example(final String str) {
        this.str = str;
    }

    public final String getName() {
        return str;
    }

    //main method
    public static void main(String[] args) {
        Example obj = new Example("Core Java Programming.");
    }
}
```

```

        System.out.println(obj.getName());
    }
}

```

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

- 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)
- Immutable classes make it easier to parallelize your program as there are no conflicts among objects.

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, a sequence of characters that can change.

The syntax of creating a StringBuffer object is:

Methods in the StringBuffer class:

- length() - used to return the length of the string i.e. total number of characters.
- reverse() - used to return the string in reversed order.
- capacity() - used to return the current capacity.

b. Write the output of the following program.

class Myoutput

```

1.  {
2.      public static void main(String args[])
3.      {
4.          String ast = "hello i love java";
5.          System.out.println(ast.indexOf('e')+" "+ast.indexOf('ast')+"
"+ast.lastIndexOf('l')+" "+ast .lastIndexOf('v'));
6.      }
7.  }

```

Output:

The program has no output

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

In the above code we have `ast.indexOf('ast')`. `indexOf()` does not take a `String` argument hence resulting to 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 run because there is an error in line 6. `"c.append(bfobj1);"`. The variable `"c"` was not created.

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

class Myoutput

```
1.  {
2.      public static void main(String args[])
3.      {
4.          StringBuffer str1 = new StringBuffer("Jambo");
5.          StringBuffer str2 = str1.reverse();
6.          System.out.println(str2);
7.      }
8.  }
```

Output: obmaJ

This is because the original `str1` having `"Jambo"` has been reversed by the `reverse()` function and transferred to the `str2` variable that is later printed.

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

class Myoutput

```
1.  {
2.      class output
3.      {
4.          public static void main(String args[])
5.          {
6.              char c[]={ 'A', '1', 'b', ' ', 'a', '0' };
7.              for (int i = 0; i < 5; ++i)
8.              {
9.                  i++;
```

```
10.         if(Character.isDigit(c[i]))
11.             System.out.println(c[i]+" is a digit");
12.         if(Character.isWhitespace(c[i]))
13.             System.out.println(c[i]+" is a Whitespace character");
14.         if(Character.isUpperCase(c[i]))
15.             System.out.println(c[i]+" is an Upper case Letter");
16.         if(Character.isLowerCase(c[i]))
17.             System.out.println(c[i]+" is a lower case Letter");
18.         i++;
19.     }
20. }
21. }
```

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.

"i" is incremented two times in the loop.