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

a. Mutable classes.

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

The mutable class examples are StringBuffer, Java. util. Date, StringBuilder.

```
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.

immutable objects are legacy classes, wrapper classes, String 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.
Mutable objects have fields that can be modified; the immutable objects have no fields that can be changed after the object is created. Immutable objects are objects whose state can not change after they have been created. Mutable and Immutable are two categories of objects in java. In this article, we will see the differences between mutable and immutable objects in java. Also, we will see java code examples showing differences between mutable and immutable class in java.

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.

StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.
The syntax of creating a StringBuffer object is:
Methods in the StringBuffer class:
-StringBuffer(): creates an empty string buffer with the initial capacity of 16.
-StringBuffer(String str): creates a string buffer with the specified string.
-StringBuffer(int capacity): creates an empty string buffer with the specified capacity as length.

a. 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

b. 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.

c. 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 of an error in line 6. “c.append(bfobj1);”. The variable “c” was not created.

d. 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.

e. 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.