1. **Write a Java program to get the character at the given index within the String**
   Code:

```
package assignment_tenth;

import java.util.Scanner;

public class CharacterAtIndex {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner scanner=new Scanner(System.in);

String str="Hello Everyone!";

System.out.println("Original String :"+str);

System.out.println("Enter the index number which you want to print:");

int n=scanner.nextInt();

System.out.println("Character at the "+n+ " Position is :"+str.charAt(n));

}

}
```

2. **Write a Java program to get the character (Unicode code point) at the given index within the String**
   Code:

```
package assignment_tenth;

import java.util.Scanner;

public class CharacterUnicode {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner scanner=new Scanner(System.in);

String str="Hello Everyone!";

System.out.println("Original String :"+str);

System.out.println("Enter the index number which you want to print:");

int n=scanner.nextInt();
```

```
System.out.println("Character Unicode at point at the "+n+" index:
"+str.codePointAt(n));

System.out.println("Character character (Unicode code point) before the "+n+"
index: "+str.codePointBefore(n));

System.out.println("Enter the ending index :");

int m=scanner.nextInt();

System.out.println("number of Unicode code points in the specified text range of
this String"+str.codePointCount(n, m));

}

}
```

3. **Write a Java program to compare two strings lexicographically. Two strings are lexicographically equal if they are the same length and contain the same characters in the same positions**
   Code:

```
package assignment_tenth;

public class CompareStringLexicographically {

public static void main(String[] args) {

// TODO Auto-generated method stub

String str1="Good";

String str2="good";

String str3="Hello";

String str4="good";

System.out.println("Comparing the strings lexicographically:
"+str1.compareTo(str2));

System.out.println("Comparing the strings lexicographically:
"+str1.compareTo(str3));

System.out.println("Comparing the strings lexicographically:
"+str1.compareToIgnoreCase(str4));

}

}
```

4. **Write a Java program to counts occurrences of a certain character in a given string**
   Code:

```
package assignment_tenth;

public class CountOccurenceOfCharacter {

public static void main(String[] args) {

// TODO Auto-generated method stub

String str1="GoodEvening";

int ch=str1.indexOf("e");

System.out.println(" occurrence of the specified character :"+ch);

int ch1=str1.indexOf("n",4);

System.out.println(" Starting the searc at specified position :"+ch1);

System.out.println("canonical representation for the string
object"+str1.intern());

System.out.println("Returns true if, and only if, length() is 0."+str1.isEmpty());

System.out.println("Returns the length of this string"+str1.length());

}

}
```

5. **Write a Java program to concatenate a given string with itself of a given number of times.**
   Code:

```
package assignment_tenth;

import java.util.Scanner;

public class ConcatStringNoOfTimes {

public static void main(String[] args) {

// TODO Auto-generated method stub

String ch="Priyanka";

System.out.println("Enter the number For contact the string");

Scanner sc =new Scanner(System.in);

int times=sc.nextInt();
```

```
StringBuilder sb = new StringBuilder();

for (int i = 0; i < times; i++) {

sb.append(ch);

}

System.out.println(sb.toString());

}

}
```

6. **Write a Java program to sort in ascending and descending order by length of the given array of strings.**

   Sample Output:
   Original unsorted colors: [Green, White, Black, Pink, Orange, Blue, Champagne, Indigo, Ivory]
   Sorted color (descending order): [Champagne, Orange, Indigo, Green, White, Black, Ivory, Pink, Blue]
   Sorted color (ascending order): [Pink, Blue, Green, White, Black, Ivory, Orange

Code:

```
package assignment_tenth;

import java.util.Arrays;

public class SortArrayByLength {

public static void main(String[] args) {

// TODO Auto-generated method stub

String[] words = {"apple", "banana", "cherry", "date", "elderberry"};

Arrays.sort(words, (a, b) -> a.length() - b.length());

System.out.println(Arrays.toString(words));

Arrays.sort(words, (a, b) -> b.length() - a.length());

System.out.println(Arrays.toString(words));

}

}
```

7. **check the given string is panlidrome or not**
   Code:

```
package assignment_tenth;

import java.util.Scanner;

public class Palidrome {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("enter a name to check plaidrome or not:");

Scanner sc=new Scanner(System.in);

String var=sc.next();

System.out.println(var);

StringBuilder sb = new StringBuilder(var);

String reversed = sb.reverse().toString();

System.out.println(reversed);

if(var.equals(reversed)) // we can't use "==" operator bcoz reversed is an object
so we have to use Default method String.equals(String)

{

System.out.println("palidrome");

}

else {

System.out.println(" not palidrome");

}

}

}
```

8. **Java Program to prove that strings are immutable in java**

Code:

```
package assignment_tenth;

public class StringImmutable {

public static void main(String[] args) {

// TODO Auto-generated method stub

String s1 = "java";

s1.concat("string is immutable");

// Yes, s1 still refers to "java",string is not change

System.out.println("s1 refers to " + s1);

}

}
```

9. **Java program to implement below classes using inheritance**
   Code:

```
package assignment_tenth.inheritance;

public class FlyingVehicles {

public static String fly(String name,String difference) {

return("Fly :"+name+":"+difference);

}

public static String land(String name,String difference) {

return("land :"+name+":"+difference);

}

}
```

```
package assignment_tenth.inheritance;

public class Airoplane extends FlyingVehicles {

private int passangers=110;
```

```
public void Fly() {
String fly=fly("airoplane","passangers");
System.out.println("Flying airplane "+fly+passangers);
}
public void land() {
String land=Land("airoplane","passangers");
System.out.println("Landing airplace "+land+passangers);
}
@Override
public String toString() {
return "Airoplane [passangers=" + passangers + "]";
}
}
```

```
package assignment_tenth.inheritance;
public class SpaceShip extends FlyingVehicles {
private boolean hyperdrive=true;
public void Fly() {
String fly=fly("spaceship","hyperdrive");
System.out.println("Flying airplane "+fly+hyperdrive);
}
public void land() {
String land=Land("spaceship","hyperdrive");
System.out.println("Landing airplace "+land+hyperdrive);
}
@Override
public String toString() {
return "SpaceShip [hyperdrive=" + hyperdrive + "]";
```

```
}

}
```

```
package assignment_tenth.inheritance;

public class GroundVehicles {

public static String drive(String name,String diff) {

return ("Drive :"+name+":"+diff);

}

}
```

```
package assignment_tenth.inheritance;

public class Car extends GroundVehicles{

private String noplates="NFS";

public void drive() {

String fly=drive("car","noplates");

System.out.println("Flying airplane "+fly+noplates);

}

public void ponderEthicalDelemma() {

System.out.println("ponderEthicalDelemma :"+noplates);

}

}
```

```
package assignment_tenth.inheritance;

public class Truck extends GroundVehicles{

private double capacity=100.0;

public void drive() {

String fly=drive("car","noplates");

System.out.println("Flying airplane "+fly+capacity);
```

```
}

public void loadCargo() {

System.out.println("load cargo :"+capacity);

}

}
```