Program 1: Write a Java program to get the character at the given index within the String

```
import java.util.Scanner;

public class Program_1 {

	public static void main(String[] args) {
		String str = new String();
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter a string:");
		str = sc.next();
		System.out.println("Enter  a number from which index u want character:");
		int n = sc.nextInt();

		System.out.println(str.charAt(n));
}
}
```

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

```
import java.util.Scanner;

public class Program_2 {

	public static void main(String[] args) {
		String str = new String();
```

```
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter a string:");
                str = sc.next();
                System.out.println("Enter  a number from which index u want character(unicode):");
                int n = sc.nextInt();
                System.out.println("Unicode of the character is:"+str.codePointAt(n));
}
}
```

Program 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

```
import java.util.Scanner;

public class Program_3{
        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);
                System.out.println("Enter two strings:");
                String string1 = sc.next();
                String string2 = sc.next();
                    System.out.println(string1.compareTo(string2));
                System.out.println("If the output is 0,it means two strings are lexicographically equal:");
                System.out.println(
                                        "If the output is lesser or greater than 0,it means two strings are
lexicographically  not equal:");

        }
```

}

Program 4: Write a Java program to counts occurrences of a certain character in a given string

```
import java.util.Scanner;

public class Program_4 {
	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter a string:");

		String string2 = sc.next();

		System.out.println(string2);

		System.out.println("enter a character");
		char ch = sc.next().charAt(0);
		int index = string2.indexOf(ch);

		int count = 0;

		for (int i = 0; i < string2.length(); i++)
			if (string2.charAt(index) == string2.charAt(i))

				count++;
		System.out.println("The occurence of a character is:" + (count++));
}
```

```
}
```

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

```
import java.util.Scanner;

public class Program_5 {

	public static void main(String[] args) {

		Scanner sc=new Scanner(System.in);

		System.out.println("Enter a string");

		String str= sc.next();

		System.out.println("enter the number of times u want to repeat the String concatenation");

		int n=sc.nextInt();

		String result = str.repeat(n);



	}

}
```

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

```
import java.util.Arrays;

import java.util.Scanner;
```

```
public class Program_6 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the number of words u want to enter:");
		int n = sc.nextInt();

		String arr[] = new String[n];
		System.out.println("Please enter the words: ");

		for (int i = 0; i < n; i++) {
			arr[i] = sc.next();

		}
		System.out.println(" Original Unsorted color:");
		for (int i = 0; i < arr.length; i++) {
			System.out.println(arr[i]);

		}
		System.out.println("-------------------------");
		System.out.println();
		Arrays.sort(arr);
		System.out.println("Sorted color (ascending order): ");
		for (int i = 0; i < arr.length; i++) {
			System.out.println(arr[i]);

		}
		System.out.print("-------------------------");
```

```
		System.out.println();
		Arrays.sort(arr);
		System.out.println("Sorted color (descending order):");
		for (int i = arr.length - 1; i >= 0; i--) {
			System.out.println(arr[i]);

		}

	}

}
```

Program 7: Write a program to check whether the given string is palindrome or not

```
public class Program_7 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter a string:");
		String str = sc.next();
		    StringBuffer sb=new StringBuffer(str);
		System.out.println(sb);
		sb.reverse();
		String revstr=sb.toString();
		System.out.println(sb);
		if(str.equals(revstr))
			System.out.println("palindrome");
		else
```

```
                System.out.println("not palindrome");

        }
```

Program 8: Java Program to prove that strings are immutable in java

```
public class Program_8 {

        public static void main(String[] args) {

                String str1 = "online classes";
                System.out.println(str1);
                String str2 = "online classes";
                str2.concat(str1);
                System.out.println(str2);
                // not allowing concatenation,still refers to the address;
                if (str1 == (str2)) {
                        System.out.println("same");
                } else
                        System.out.println("Not same");
                System.out.println("--------------------");

                String name1 = new String("Edubridge");// object
                String name2 = new String("Edubridge");
                if (name1 == name2) {
```

```
                System.out.println(" Both are pointing to same reference");
        } else
                System.out.println(" Both are pointing to different reference");
        // though the values are same both are @ different address and are the objects
        // are not same,which means strings are immutable

        name1.concat("java");// when we try to modify the string existing value,its not changing
,the
                                                        // reference variable still points the
existing value,because strings are
                                                        // immutable
        System.out.println(name1);
        name1 = name1 + "java class";// if we want the reference variable to point newly
assigned value then we
                                                                // should explicitly
assign it to the variable;
        System.out.println(name1);

    }

}
```

Program 9: Java program to implement the concept of inheritance

```
public class Flyingvehicle {
    Flyingvehicle() {
        System.out.println("Flying vehicle!!!!!");
    }

    void fly() {
```

```
		System.out.println("Flying vehicle Take off from land and fly in the sky");
	}

	void land() {
		System.out.println("Flying vehicle land in resevered place like
helipad,airport,spacestation etc.,");

	}

}

public class Spaceship extends Flyingvehicle {
	Spaceship() {
		System.out.println("Spaceship!!!");
	}

	boolean hyperdrive;

	void fly() {
		// super.fly();
		System.out.println("Spacecraft use rockets to take off from the earth:");

	}

	void land() {
		System.out.println("The spacecrafts are landed like a glider airplane ");

	}
```

```
}


import java.util.Scanner;

public class Airplane extends Flyingvehicle {
	Airplane() {
		System.out.println("Airplane");
	}

	void fly() {
		Scanner sc = new Scanner(System.in);
		System.out.println("enter the number of passengers:");

		int passengers = sc.nextInt();

		if (passengers > 600)
			System.out.println("cannot fly,seats are limite to 600 only!");
		else
			System.out.println("we are ready to fly");

		System.out.println();
		// System.out.println("Airplane uses its built wings just like birds wings to
		// fly high which are operated by pilot");
	}

	void land() {
		super.land();
```

```
		System.out.println("Pilot opens the flats and slats to make the wings"
				+ " bigger and the airbrakes to lower the planes and landing gear to land airplane");

	}
}


public class Groundvehicle {
	Groundvehicle() {
		System.out.println("Ground vehicle!!!");
	}

	void drive() {
		System.out.println("Ground vehicles are driven on roads and many individuals own ground vehicles");

	}

}


public class Car extends Groundvehicle {
	Car() {
		super();
	}

	int noPlates = 6;
```

```
	void drive() {
		System.out.println("There are " + noPlates + " types of number plates of cars");
		System.out.println("cars  have automatic gears too ");
		System.out.println("Maximum number of gears car can have is 12");

	}

	void ponderEthicalDilemma() {
		System.out.println("Should i follow the traffic rules" + " or should i skip the skip the
signal");

		System.out.println("should i go left or right:");
	}

}


import java.util.Scanner;

public class Truck extends Groundvehicle {
	Truck() {
		super();
	}

	void drive() {
		System.out.println("In India most of the trucks are manual shift gears");
		System.out.println("Trucks have  18 gears");
	}
```

```
	void loadCargo() {

		System.out.println("Enter how many kilogram of cargo u want to load");
		Scanner sc = new Scanner(System.in);
		double capacity = sc.nextDouble();
		if (capacity > 55000)
			System.out.println("Cannot load cargo of weight " + capacity);
		else
			System.out.println("Good to go");
	}

}


public class TestVehicle {

	public static void main(String[] args) {
		Flyingvehicle flyingvehicle = new Flyingvehicle();
		flyingvehicle.fly();
		flyingvehicle.land();
		Flyingvehicle flyingvehicle1 = new Spaceship();// upcasting
		flyingvehicle1.fly();
		flyingvehicle1.land();
		Airplane airplane = new Airplane();
		airplane.fly();
		airplane.land();
		System.out.println("-----------------------------------------------------");
		Groundvehicle groundvehicle = new Groundvehicle();
```

```
        groundvehicle.drive();
        Groundvehicle groundvehicle1 = new Truck();
        groundvehicle1.drive();
        Truck truck = new Truck();
        truck.loadCargo();
        Car car = new Car();
        car.drive();
        car.ponderEthicalDilemma();

    }

}
```

Program 10:Attendance Management

```
public class Person {
    private String id;
    private String name;
    private String password;
    private String email;

    public Person(String id, String name, String password, String email) {
        super();
        this.id = id;
        this.name = name;
        this.password = password;
```

```
        this.email = email;
}

public String getId() {
        return id;
}

public void setId(String id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getPassword() {
        return password;
}

public void setPassword(String password) {
        this.password = password;
}

public String getEmail() {
        return email;
```

```
	}

	public void setEmail(String email) {
		this.email = email;
	}

	@Override
	public String toString() {
		return "Person [id=" + id + ", name=" + name + ", password=" + password + ", email=" +
email + "]";
	}


}



import java.util.Arrays;

public class Admin extends Person {
	static Teachers[] teacherList=new Teachers[15];
	//add coursearray
	static Course[] courseList=new Course[15];
	//add studentarray
	static Students[] studentList=new Students[15];
	static int count=0;
	static int countS=0;
	static int countC=0;
	//count for course and student
```

```
	public Admin(String id, String name, String password, String email) {
		super(id, name, password, email);
		// TODO Auto-generated constructor stub
	}


public static Teachers[] getTeacherList() {
		return teacherList;
	}


	public static void setTeacherList(Teachers[] teacherList) {
		Admin.teacherList = teacherList;
	}


	public static Course[] getCourseList() {
		return courseList;
	}


	public static void setCourseList(Course[] courseList) {
		Admin.courseList = courseList;
	}


	public static Students[] getStudentList() {
		return studentList;
	}
```

```
public static void setStudentList(Students[] studentList) {
	Admin.studentList = studentList;
}

public static int getCount() {
	return count;
}

public static void setCount(int count) {
	Admin.count = count;
}

public static int getCountS() {
	return countS;
}

public static void setCountS(int countS) {
	Admin.countS = countS;
}

public static int getCountC() {
	return countC;
```

```
	}



	public static void setCountC(int countC) {
		Admin.countC = countC;
	}



@Override
	public String toString() {
		return super.toString()+"Admin [teacherList=" + Arrays.toString(teacherList) + "]";
	}

public void addNewTeacher(Teachers teacher)
{
	teacherList[count++]=teacher;//0--10000,1--20000,2--30000


}
public void viewTeacherList()
{
	for(int i =0;i<count;i++)//1<3
	{
		System.out.println("teacher list  : "+teacherList[i]);
	}
}
public void modifyTeacherInfo(String id,String password)
{
	for(int i=0;i<count;i++)//325
	{
```

```
                if(teacherList[i].getId().equals(id))
                {
                        teacherList[i].setPassword(password);
                        break;
                }
        }

}
public void removeTeacherById(String id)
{   int pos=-1;
        for(int i=0;i<count;i++)
        {
                if(teacherList[i].getId().equals(id))
                {
                        pos= i;
                        break;
                }
        }
        for(int i=pos;i<count;i++)
        {
                teacherList[i] = teacherList[i+1];

        }
        if(pos>=0)
        {
                count--;
        }
        }
//implement add student,course
```

```
public void AddNewStudents(Students student) {
		studentList[countS++] = student;
	}

	public void viewStudentList() {
		for (int i = 0; i < countS; i++) {
			System.out.println("student list  : " + studentList[i]);
		}
	}

	public void modifyStudentInfo(String id, String password) {
		for (int i = 0; i < countS; i++) {
			if (studentList[i].getId().equals(id)) {
				studentList[i].setPassword(password);
				break;
			}
		}

	}

	public void removeStudentById(String id) {
		int pos = -1;
		for (int i = 0; i < countS; i++) {
			if (studentList[i].getId().equals(id)) {
				pos = i;
				break;
			}
		}
		for (int i = pos; i < countS; i++) {
```

```
                studentList[i] = studentList[i + 1];


        }
        if (pos >= 0) {
                countS--;
        }
}
//implement course add and modify
public void AddNewCourse(Course course) {
        courseList[countC++] = course;
}

public void viewCourseList() {
        for (int i = 0; i < countC; i++) {
                System.out.println("course list  : " + courseList[i]);
        }
}

public void modifyCourseInfo(String id, String teacher) {
        for (int i = 0; i < countC; i++) {
                if (courseList[i].getId().equals(id)) {
                        courseList[i].setTeacher(teacher);
                        break;
                }
        }

}

public void removeCourseById(String id) {
```

```
                int pos = -1;
                for (int i = 0; i < countC; i++) {
                        if (courseList[i].getId().equals(id)) {
                                pos = i;
                                break;
                        }
                }
                for (int i = pos; i < countC; i++) {
                        courseList[i] = courseList[i + 1];

                }
                if (pos >= 0) {
                        countC--;
                }
        }
//implement update student and course
        public void assignCouse(String id,Course course) {
                for(int i=0;i<countS;i++) {
                        if(studentList[i].getId().equals(id)) {
                                studentList[i].setCourse(course);
                        }
                }
        }
//implement delete student and course

}


public class Teachers extends Person {
```

```
	Students[] s=new Students[15];
	public Teachers(String id, String name, String password, String email) {
		super(id, name, password, email);
		// TODO Auto-generated constructor stub
	}
	//mark attendance method
	public void MarkAttendance(String id,boolean attendance) {
		s=Admin.getStudentList();
		for(int i=0;i<Admin.countS;i++) {
		if(s[i].getId().equals(id)) {
			Admin.getStudentList()[i].setAttendance(attendance);
		}
		}
	}

}



public class Students  extends Person{
	String id,name, password, email;
	Course course;
	boolean attendance;
	public Students(String id,String name, String password, String email,Course course) {
		super(id, name, password, email);
		this.id=id;
		this.name=name;
		this.password=password;
		this.email=email;
		this.course=course;
```

```
	}
	public Course getCourse() {
		return course;
	}
	public void setCourse(Course course) {
		this.course = course;
	}
	public boolean isAttendance() {
		return attendance;
	}
	public void setAttendance(boolean attendance) {
		this.attendance = attendance;
	}
	@Override
	public String toString() {
		return "Students [id=" + id + ", name=" + name + ", password=" + password + ", email="
+ email + ", course="
				+ course.getName() + ", attendance=" + attendance + "]";
	}
}
```

```
import java.util.Arrays;

public class Course {
```

```
private String id,name,teacher;
private String[] eStudents=new String[15];
public Course(String id,String name,String teacher,String[] eStudents) {
        super();
        this.id=id;
        this.name=name;
        this.teacher=teacher;
        this.eStudents=eStudents;
}
public String getId() {
        return id;
}
public void setId(String id) {
        this.id = id;
}
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getTeacher() {
        return teacher;
}
public void setTeacher(String teacher) {
        this.teacher = teacher;
}
public String[] geteStudents() {
        return eStudents;
```

```
	}
	public void seteStudents(String[] eStudents) {
		this.eStudents = eStudents;
	}

	@Override
	public String toString() {
		return "Course [id=" + id + ", name=" + name + ", teacher=" + teacher + ", eStudents="
				+ Arrays.toString(eStudents)  + "]";
	}

}



import java.io.IOException;

public class TestAttendance {



		// TODO Auto-generated method stub
		public static void main(String[] args) throws IOException {
			// TODO Auto-generated method stub
			Admin admin = new Admin("1234", "Harish", "hari@123",
"harish@gmail.com");
			Teachers teacher = new Teachers("1234", "Indu", "indu#12",
"indu@gmail.com");
			// Teacher
			admin.addNewTeacher(new Teachers("324", "Indu", "indu#12",
"indu@gmail.com"));
```

```
admin.addNewTeacher(new Teachers("325", "raj", "raj#12", "raj@gmail.com"));
admin.addNewTeacher(new Teachers("326", "Kalyan", "k#12", "k@gmail.com"));

admin.viewTeacherList();

System.out.println("After modification");
admin.modifyTeacherInfo("325", "raj@123");
admin.viewTeacherList();

System.out.println("After removal");
admin.removeTeacherById("325");
admin.viewTeacherList();

// Course
System.out.println("Adding Course");
String[] studentList = { "Arshad", "world" };
admin.AddNewCourse(new Course("13", "Science", "Indu", studentList));
admin.AddNewCourse(new Course("11", "world", "Kalyan", studentList));

admin.viewCourseList();

System.out.println("After Modification");
admin.modifyCourseInfo("13", "World");
admin.viewCourseList();

System.out.println("After Removal");
admin.removeCourseById("11");
admin.viewCourseList();
```

```
                        // Student
                        System.out.println("Adding student");
                        admin.AddNewStudents(new Students("134", "Arshad", "varad@123",
"varad@gmail.com",new Course("13", "Science", "Indu", studentList)));
                        admin.AddNewStudents(new Students("11", "world", "world@123",
"world@gmail.com",new Course("13", "Science", "Indu", studentList)));

                        admin.viewStudentList();

                        System.out.println("After modification");
                        admin.modifyStudentInfo("134", "raj@123");
                        admin.viewStudentList();

                        System.out.println("After removal");
                        admin.removeStudentById("134");
                        admin.viewStudentList();

                        // assign course
                        System.out.println("Assign course to student");
                        admin.assignCouse("11", new Course("13", "Science", "Indu", studentList));
                        admin.viewStudentList();

                        // setAttendance
                        System.out.println("Assign attendance");
                        teacher.MarkAttendance("11", true);
                        admin.viewStudentList();

                }
        }
```