1)Write a Java program to get the character at the given index within the String.

ANS) public class CharacterIndex {

public static void main(String[] args) {

String str = "program";

int index = 4;

System.out.println("Character at index " + index + ": " + str.charAt(index));

}

}

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

Ans)

public class UnicodeAtIndex {

public static void main(String[] args) {

String str = "FULLSTACK";

int index = 2;

int codePoint = str.codePointAt(index);

System.out.println("Unicode code point at index " + index + ": " + codePoint);

}

}

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

ANS)public class CompareString {

public static void main(String[] args) {

String str1 = "RED";

String str2 = "BLUE";

int comparison = str1.compareTo(str2);

```
        if (comparison < 0) {
            System.out.println(str1 + " is lexicographically less than " + str2);
        } else if (comparison > 0) {
            System.out.println(str1 + " is lexicographically greater than " + str2);
        } else {
            System.out.println(str1 + " is lexicographically equal to " + str2);
        }
    }
}
```

4)Write a Java program to counts occurrences of a certain character in a given string.

ANS)

```
public class CountCharacterOccurrences {
    public static void main(String[] args) {
        String str = "good";
        char target = 'o';
        long count = str.chars().filter(ch -> ch == target).count();
        System.out.println("Occurrences of '" + target + "': " + count);
    }
}
```

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

ANS)

```
public class StringCon {

    public static void main(String[] args) {
        String st = "Hello";
        int times = 2;
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < times; i++) {
            result.append(st);
        }

        System.out.println("Resultant string: " + result.toString());
```

```
    }
}
```

7)check the given string is panlidrome or not.

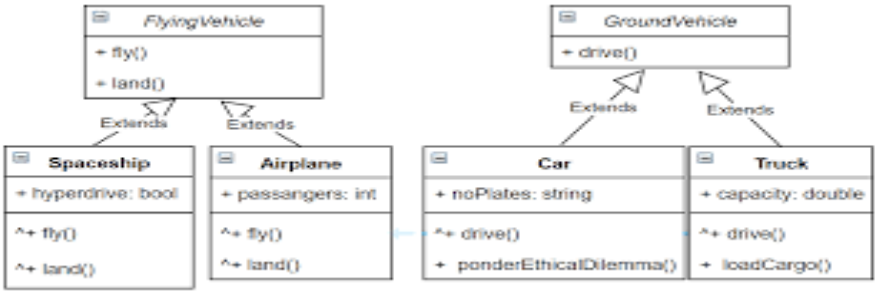
ANS)

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "wow";
        String reversed = new StringBuilder(str).reverse().toString();

        if (str.equals(reversed)) {
            System.out.println(str + " is a palindrome.");
        } else {
            System.out.println(str + " is not a palindrome.");
        }
    }
}
```

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

```
public class StringImmutability {
    public static void main(String[] args) {
        String str = "Hello";
        String str1 = str.concat(", World!");

        System.out.println("Original string: " + str);
        System.out.println("Modified string: " + str1);
    }
}
```

Java program to implement below classes using inheritance

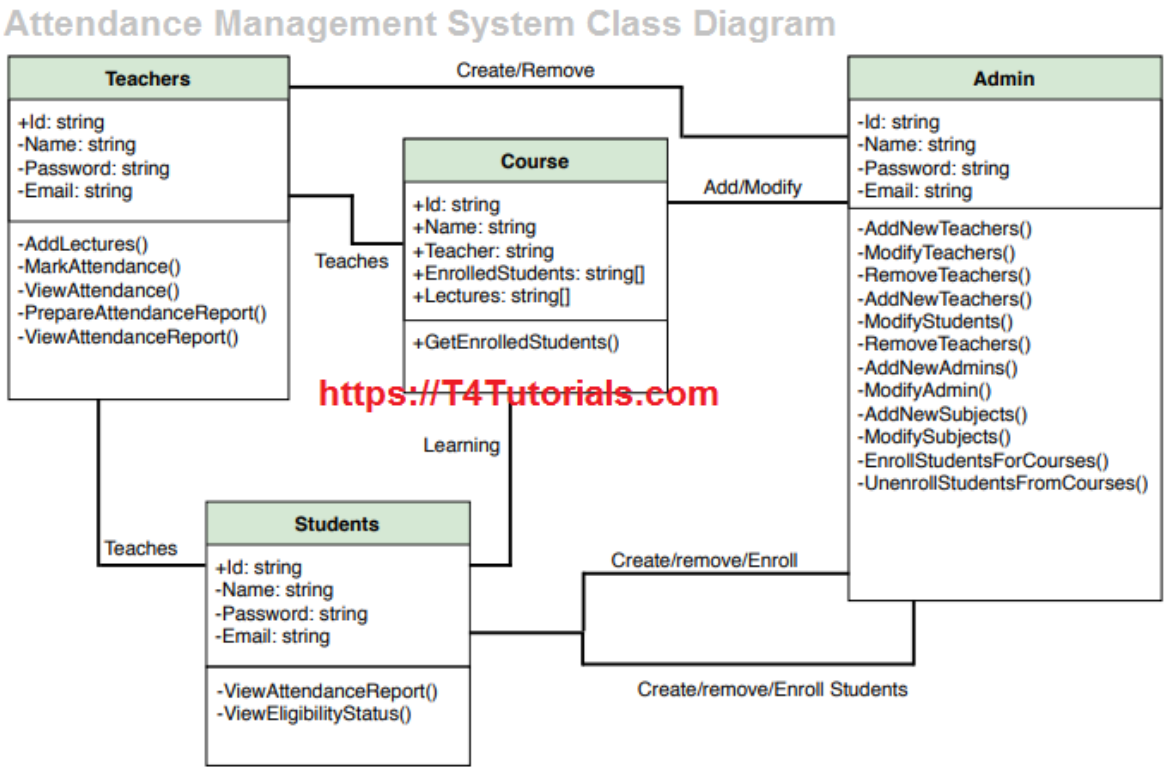



```
class FlyingVehicle {
    public void fly() {
        System.out.println("Flying");
    }

    public void land() {
        System.out.println("Landing");
    }
}

class GroundVehicle {
    public void drive() {
        System.out.println("Driving");
    }
}

class Spaceship extends FlyingVehicle {
    private boolean hasHyperdrive;

    public Spaceship(boolean hasHyperdrive) {
        this.hasHyperdrive = hasHyperdrive;
    }

    public void fly() {
        if (hasHyperdrive) {
```

```
            System.out.println("Hyperspace jump!");
        } else {
            super.fly();
        }
    }
}

class Airplane extends FlyingVehicle {
    private int passengers;

    public Airplane(int passengers) {
        this.passengers = passengers;
    }

    public void fly() {
        System.out.println("Airplane flying with " + passengers + " passengers.");
    }
}

class Car extends GroundVehicle {
    private String noPlates;

    public Car(String noPlates) {
        this.noPlates = noPlates;
    }

    public void drive() {
        System.out.println("Car driving with plate number " + noPlates);
    }
}
```

```
class Truck extends GroundVehicle {
    private double capacity;

    public Truck(double capacity) {
        this.capacity = capacity;
    }

    public void drive() {
        System.out.println("Truck driving with capacity " + capacity);
    }

    public void loadCargo() {
        System.out.println("Loading cargo...");
    }
}
```

1. Write a java program to implement the below diagram




```
class Teacher {
```

```
    private String id;
    private String name;
    private String password;
    private String email;

    // ... other methods
}

class Admin extends Teacher {
    public void addNewTeachers() {
        // ...
    }

    public void modifyTeachers() {
        // ...
    }

    public void removeTeachers() {
        // ...
    }

    // ... other methods
}

class Course {
    private String id;
    private String name;
    private Teacher teacher;
    private String[] enrolledStudents;

    // ... other methods
```

```
}

class Student {
    private String id;
    private String name;
    private String password;
    private String email;

    // ... other methods
}

class Learning {
    public void enrollStudentsForCourses() {
        // ...
    }

    public void unenrollStudentsFromCourses() {
        // ...
    }

    // ... other methods
}
```