NAME:DEBAGNI BHATTACHARJEE **Roll No.:** 04
Enrollment No.: 12023006015004 **SEC:** A **Semester:** 2nd
DEPARTMENT: Department of Computer Application(MCA)

Week: 6

1. Design an abstract class having two methods. Create Rectangle and Triangle classes by inheriting the shape class and override the above methods to suitably implement for Rectangle and Triangle class.

Code:

```
abstract class Shape {
    public abstract double calculateArea();
    public abstract double calculatePerimeter();
}
class Rectangle extends Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double calculateArea() {
        return length * width;
    }
    public double calculatePerimeter() {
        return 2 * (length + width);
    }
}
class Triangle extends Shape {

private double side1;  private
    double side2;
    private double side3;
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }
    public double calculateArea() {
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));

    }

    public double calculatePerimeter() {
        return side1 + side2 + side3;
    }
}
public class ShapeTest {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 4);
        Triangle triangle = new Triangle(3, 4, 5);
        System.out.println("Rectangle Area: " + rectangle.calculateArea());
        System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());
        System.out.println("Triangle Area: " + triangle.calculateArea());
        System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());
    }
}
```

```
Rectangle Area: 20.0
Rectangle Perimeter: 18.0
Triangle Area: 6.0
Triangle Perimeter: 12.0
PS C:\Users\User\Desktop\Java\practice>
```

2. Write a program in Java to illustrate the use of interface in Java.

Code:

```
interface Printable {
    void print();
}
class Printer implements Printable {


    public void print() {
        System.out.println("Printing...");
    }
}
public class InterfaceExample {
    public static void main(String[] args) {
        Printable printer = new Printer();
        printer.print();
    }
}
```

3. Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea ( ) and volume( ). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three-dimensional objects. The dimensions of the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

Code:

```
abstract class ThreeDObject {
    public abstract double wholeSurfaceArea();
    public abstract double volume();
}
class Box extends ThreeDObject {
    private double length;
    private double width;
    private double height;
    public Box(double length, double width, double height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }
    public double wholeSurfaceArea() {
        return 2 * (length * width + length * height + width * height);
    }
    public double volume() {
        return length * width * height;
    }
}
class Cube extends ThreeDObject {
    private double side;
    public Cube(double side) {
```

```
    public double wholeSurfaceArea() {
        return 6 * side * side;
    }
    public double volume() {
        return side * side * side;
    }
}
class Cylinder extends ThreeDObject {
    private double radius;
    private double height;
    public Cylinder(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }
    public double wholeSurfaceArea() {
        return 2 * Math.PI * radius * (radius + height);
    }
    public double volume() {
        return Math.PI * radius * radius * height;
    }
}
class Cone extends ThreeDObject {
    private double radius;
    private double height;
    public Cone(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }
    public double wholeSurfaceArea() {

        double slantHeight = Math.sqrt(radius * radius + height * height);
        return Math.PI * radius * (radius + slantHeight);
     public double volume() {
        return (1.0 / 3.0) * Math.PI * radius * radius * height;
    }

}
public class ThreeDObjectTest {
    public static void main(String[] args) {
        Box box = new Box(2, 3, 4);
        System.out.println("Box Whole Surface Area: " + box.wholeSurfaceArea());
        System.out.println("Box Volume: " + box.volume());
        Cube cube = new Cube(5);
        System.out.println("Cube Whole Surface Area: " + cube.wholeSurfaceArea());
        System.out.println("Cube Volume: " + cube.volume());
        Cylinder cylinder = new Cylinder(3, 6);
        System.out.println("Cylinder Whole Surface Area: " + cylinder.wholeSurfaceArea());
        System.out.println("Cylinder Volume: " + cylinder.volume());
        Cone cone = new Cone(4, 7);
        System.out.println("Cone Whole Surface Area: " + cone.wholeSurfaceArea()); }
```

```
Box Whole Surface Area: 52.0
Box Volume: 24.0
Cube Whole Surface Area: 150.0
Cube Volume: 125.0
Cylinder Whole Surface Area: 188.4955592153876
Cylinder Volume: 169.64600329384882
Cone Whole Surface Area: 175.92918860102842
Cone Volume: 117.6470588235294
PS C:\Users\User\Desktop\Java\practice>
```

4. Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData ( ) to show "This is a vehicle class". Inherit the Vehicle class into subclasses named Bus and Car having individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData ( ) method showing all details of Bus and Car respectively with content of the super class's showData ( ) method.

Code:

```
class Vehicle {
    protected String regnNumber;
    protected double speed;
    protected String color;

    protected String ownerName;
    public Vehicle(String regnNumber, double speed, String color, String ownerName) {
        this.regnNumber = regnNumber;
        this.speed = speed;
        this.color = color;
        this.ownerName = ownerName;
    }
    protected void showData() {
        System.out.println("This is a vehicle class");
    }
}
class Bus extends Vehicle {
    private int routeNumber;
    public Bus(String regnNumber, double speed, String color, String ownerName, int

    routeNumber) {
        super(regnNumber, speed, color, ownerName);
        this.routeNumber = routeNumber;
    }
    protected void showData() {
        super.showData();
        System.out.println("Route Number: " + routeNumber);
    }
    }
class Car extends Vehicle {
    private String manufacturerName;
    public Car(String regnNumber, double speed, String color, String ownerName, String
manufacturerName) {
        super(regnNumber, speed, color, ownerName);
        this.manufacturerName = manufacturerName;
    }
    protected void showData() {
        super.showData();
        System.out.println("Manufacturer Name: " + manufacturerName);

    }
}
public class VehicleTest {
    public static void main(String[] args) {
        Bus bus = new Bus("ABC123", 60.0, "Red", "John Doe", 101);
        bus.showData();
        Car car = new Car("XYZ456", 100.0, "Blue", "Jane Doe", "Toyota");
        car.showData();
    }
}
```

```
This is a vehicle class
Route Number: 101
This is a vehicle class
Manufacturer Name: Toyota
```

5. Create three interfaces, each with two methods. Inherit a new interface from the three,
adding a new method. Create a class by implementing the new interface and also inheriting from a concrete class. Now write four methods, each of which takes one of the
four interfaces as an argument. In main ( ), create an object of your class and pass it to
each of the methods.

Code:

```
interface Interface1 {
    void method1();
    void method2();
}
interface Interface2 {
    void method3();
    void method4();
}
interface Interface3 extends Interface1, Interface2 {
    void method5();
}
class MyClass implements Interface3 {


    public void method1() {
        System.out.println("Method 1");
    }
    public void method2() {
        System.out.println("Method 2");
    }
    public void method3() {
        System.out.println("Method 3");
    }
    public void method4() {
        System.out.println("Method 4");

  }

  public void method5() {

     System.out.println("Method 5");

  }

}

public class InterfaceInheritanceTest {

  public static void main(String[] args) {

     MyClass myObj = new MyClass();

     myObj.method1();

     myObj.method2();

     myObj.method3();

     myObj.method4();
```

```
Method 1
Method 2
Method 3
Method 4
Method 5
```

6. Create an interface Department containing attributes deptName and deptHead. It also has abstract methods for printing the attributes. Create a class hostel containing hostelName, hostelLocation and numberofRooms. The class contains methods for getting and printing the attributes. Then write Student class extending the Hostel class and implementing the Department interface. This class contains attributes studentName, regdNo, electiveSubject and avgMarks. Write suitable getData and printData methods for this class. Also implement the abstract methods of the Department interface. Write a driver class to test the Student class. The program should be menu driven containing the options:
i) Admit new student
ii) Migrate a student
iii) Display details of a student
For the third option a search is to be made on the basis of the entered registration number.

Code:

```
interface Department {
    void printDepartment();
}
class Hostel {
    protected String hostelName;
    protected String hostelLocation;
    protected int numberOfRooms;
    public Hostel(String hostelName, String hostelLocation, int numberOfRooms) {
        this.hostelName = hostelName;
        this.hostelLocation = hostelLocation;
        this.numberOfRooms = numberOfRooms;
    }
    public void printHostel() {
        System.out.println("Hostel Name: " + hostelName);
        System.out.println("Hostel Location: " + hostelLocation);
        System.out.println("Number of Rooms: " + numberOfRooms);
    }
}
class Student extends Hostel implements Department {
    private String studentName;
    private int regdNo;
    private String electiveSubject;
    private double avgMarks;
    public Student(String hostelName, String hostelLocation, int numberOfRooms, String
studentName, int regdNo, String electiveSubject, double avgMarks) {
        super(hostelName, hostelLocation, numberOfRooms);
        this.studentName = studentName;
        this.regdNo = regdNo;
        this.electiveSubject = electiveSubject;
        this.avgMarks = avgMarks;
    }
```

```
    public void printDepartment() {
        System.out.println("Student Department Information:");
        System.out.println("Student Name: " + studentName);
        System.out.println("Registration Number: " + regdNo);
        System.out.println("Elective Subject: " + electiveSubject);
        System.out.println("Average Marks: " + avgMarks);
    }
}
public class ClassHierarchyTest {
    public static void main(String[] args) {


        Student student = new Student("ABC Hostel", "XYZ Location", 100, "John Doe", 12345,
"Mathematics", 85.5);
        student.printHostel();
        student.printDepartment();
    }
}
```

```
Hostel Name: ABC Hostel
Hostel Location: XYZ Location
Number of Rooms: 100
Student Department Information:
Student Name: John Doe
Registration Number: 12345
Elective Subject: Mathematics
Average Marks: 85.5
```

7. Create an interface called Player. The interface has an abstract method called play() that displays a message describing the meaning of "play" to the class. Create classes called Child, Musician, and Actor that all implement Player. Create an application that demonstrates the use of the classes(UsePlayer.java

Code:

```
interface Player {
    void play();
}
class Child implements Player {
    public void play() {
        System.out.println("Child is playing with toys.");
    }
}
class Musician implements Player {
    public void play() {
        System.out.println("Musician is playing an instrument.");
    }
}
class Actor implements Player {
    public void play() {
        System.out.println("Actor is performing in a play.");
    }
}
public class PlayerTest {
    public static void main(String[] args) {
        Player child = new Child();
        Player musician = new Musician();

        Player actor = new Actor();
        child.play();
        musician.play();
        actor.play();
```

```
}
```

```
Child is playing with toys.
Musician is playing an instrument.
Actor is performing in a play.
```

8. Create an abstract class Accounts with the following details:
Data Members:
(a) Balance (b) accountNumber (c) accountHoldersName (d) address
Methods:
(a) withdrawl()- abstract
(b) deposit()- abstract
(c) display() to show the balance of the account number
Create a subclass of this class SavingsAccount and add the following details:

Data Members:
(a) rateOfInterest
Methods:
(a) calculateAount()

Code:

```
abstract class Accounts {
    protected double balance;
    protected int accountNumber;
    protected String accountHoldersName;
    protected String address;

    public abstract void withdrawl();
    public abstract void deposit();

    public void display() {

        System.out.println("Balance of Account Number " + accountNumber + ": " + balance);
    }
}
class SavingsAccount extends Accounts {
    private double rateOfInterest;
    public SavingsAccount(double balance, int accountNumber, String accountHoldersName,
String address, double rateOfInterest) {
        this.balance = balance;
        this.accountNumber = accountNumber;
        this.accountHoldersName = accountHoldersName;
        this.address = address;
        this.rateOfInterest = rateOfInterest;
    }
    public void calculateAmount() {
        balance += balance * (rateOfInterest / 100);
    }
}
public class AccountsTest {
    public static void main(String[] args) {
        SavingsAccount savingsAccount = new SavingsAccount(1000, 12345, "John Doe",
"123 Main St", 5.0);
        savingsAccount.deposit();
        savingsAccount.withdrawl();
        savingsAccount.calculateAmount();
        savingsAccount.display();
    }
}
```

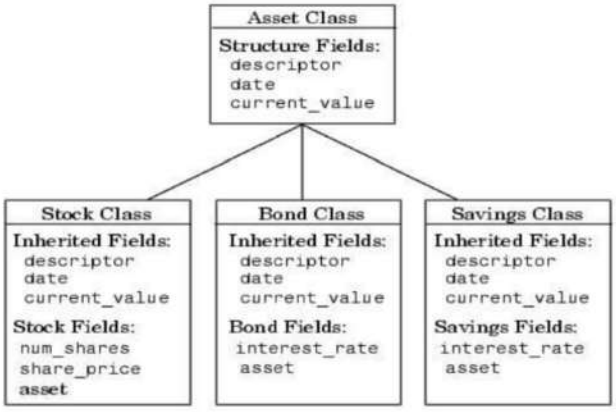
9. Create an abstract class MotorVehicle with the following details:
Data Members:
(a) modelName (b)modelNumber (c) modelPrice
Methods:
(a) display() to show all the details
Create a subclass of this class Carthat inherits the class MotorVehicle and add the following details:
Data Members:

(b) discountRate
Methods:
(a) display() method to display the Car name, model number, price and the discount rate.
(b) discount() method to compute the discount.

Code:
```
abstract class MotorVehicle {
    protected String modelName;
    protected int modelNumber;
    protected double modelPrice;
    public void display() {
        System.out.println("Model Name: " + modelName);
        System.out.println("Model Number: " + modelNumber);
        System.out.println("Model Price: " + modelPrice);
    }
}
class Car extends MotorVehicle {
    private double discountRate;
    public Car(String modelName, int modelNumber, double modelPrice, double
discountRate) {

        this.modelName = modelName;
        this.modelNumber = modelNumber;
        this.modelPrice = modelPrice;
        this.discountRate = discountRate;
    }
    public void display() {
        super.display();
        System.out.println("Discount Rate: " + discountRate);
    }
}
public class MotorVehicleTest {
    public static void main(String[] args) {
        Car car = new Car("Toyota", 123, 25000, 10.0);
        car.display();
    }


}
```

```
Model Name: Toyota
Model Number: 123
Model Price: 25000.0
Discount Rate: 10.0
```

10. Implement the below Diagram.
Here, Asset class is an abstract class containing an abstract method displayDetails() method. Stock, bond and Savings class inherit the Asset class and displayDetails() method is defined in every class.



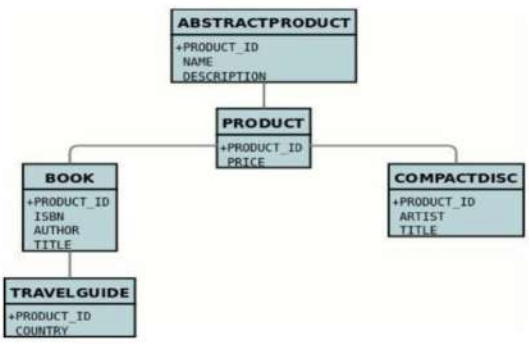
Code: abstract class Asset {

```
    private String descriptor;

    private String date;

    private double currentValue;

    public Asset(String descriptor, String date, double currentValue) {

        this.descriptor = descriptor;

        this.date = date;

        this.currentValue = currentValue;

    }


    public abstract void displayDetails();
}
class Stock extends Asset {
    private int numShares;
    private double sharePrice;
    public Stock(String descriptor, String date, double currentValue, int numShares, double
sharePrice) {

        super(descriptor, date, currentValue);
        this.numShares = numShares;
        this.sharePrice = sharePrice;
    }
    public void displayDetails() {
        System.out.println("Asset: Stock");
        System.out.println("Descriptor: " + descriptor);
        System.out.println("Date: " + date);
        System.out.println("Current Value: $" + currentValue);
        System.out.println("Number of Shares: " + numShares);
        System.out.println("Share Price: $" + sharePrice);
    }
}
class Bond extends Asset {
    private String interestRate;
    private double asset;
    public Bond(String descriptor, String date, double currentValue, String interestRate,
double asset) {
```

```
        super(descriptor, date, currentValue);
        this.interestRate = interestRate;
        this.asset = asset;
    }
    public void displayDetails() {
        System.out.println("Asset: Bond");
        System.out.println("Descriptor: " + descriptor);
        System.out.println("Date: " + date);
        System.out.println("Current Value: $" + currentValue);
        System.out.println("Interest Rate: " + interestRate);
        System.out.println("Asset Value: $" + asset);
    }
}
class Savings extends Asset {
    private String interestRate;
    public Savings(String descriptor, String date, double currentValue, String interestRate) {

        super(descriptor, date, currentValue);
        this.interestRate = interestRate;
    }
    public void displayDetails() {
        System.out.println("Asset: Savings");
        System.out.println("Descriptor: " + descriptor);
        System.out.println("Date: " + date);
        System.out.println("Current Value: $" + currentValue);
        System.out.println("Interest Rate: " + interestRate);
    }
}
```

```
Stock Details:
Asset: Stock
Descriptor: Apple Inc.
Date: 2023-03-15
Current Value: $12575.0
Number of Shares: 100
Share Price: $125.75

Bond Details:
Asset: Bond
Descriptor: US Treasury Bond
Date: 2022-09-01
Current Value: $10000.0
Interest Rate: 3.5%
Asset Value: $9850.0

Savings Details:
Asset: Savings
Descriptor: Savings Account
Date: 2021-06-30
Current Value: $25000.0
Interest Rate: 2.75%
```

11. Implement the below Diagram. Here AbstractProduct is only abstract class.



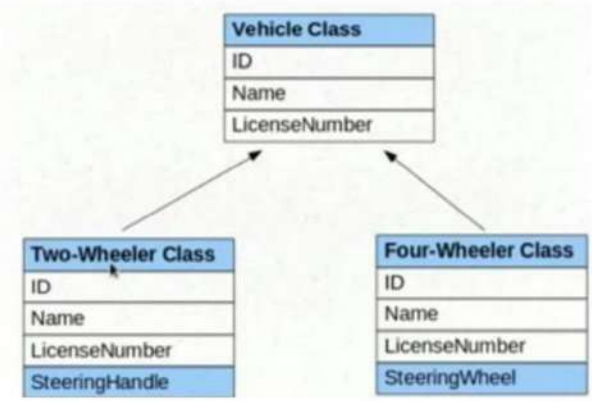
Code: abstract class AbstractProduct {

```
    private int productId;
    private String description;
    public AbstractProduct(int productId, String description) {
        this.productId = productId;
        this.description = description;
    }
    public int getProductId() {
        return productId;
    }
    public String getDescription() {
        return description;
    }
    public abstract void showDetails();
}
class Product extends AbstractProduct {
    private double price;
    public Product(int productId, String description, double price) {
        super(productId, description);
        this.price = price;
    }
    public double getPrice() {
        return price;

    }
    public void showDetails() {
        System.out.println("Product ID: " + getProductId());
        System.out.println("Description: " + getDescription());
        System.out.println("Price: $" + price);
    }
}
class Book extends Product {
    private String author;
    private String title;
    public Book(int productId, String description, double price, String author, String title) {
        super(productId, description, price);
        this.author = author;
        this.title = title;
    }
    public void showDetails() {
        super.showDetails();
        System.out.println("Author: " + author);
        System.out.println("Title: " + title);
    }
}
class TravelGuide extends Book {
    private String location;
    public TravelGuide(int productId, String description, double price, String author, String title,
String location) {
        super(productId, description, price, author, title);
        this.location = location;
    }
    public void showDetails() {
        super.showDetails();
        System.out.println("Location: " + location);
    }
}
```

```
class CompactDisc extends Product {
    private String artist;
    public CompactDisc(int productId, String description, double price, String artist) {
        super(productId, description, price);
        this.artist = artist;
    }
    public void showDetails() {
        super.showDetails();
        System.out.println("Artist: " + artist);
    }
}
```

```
Book Details:
Product ID: 1
Description: Fiction Novel
Price: $14.99
Author: John Doe
Title: The Lost Kingdom

Travel Guide Details:
Product ID: 2
Description: Travel Guide
Price: $19.99
Author: Jane Smith
Title: Explore Italy
Location: Rome

Compact Disc Details:
Product ID: 3
Description: Pop Album
Price: $9.99
Artist: The Melodies
```

12. Implement the below Diagram



Code: class Vehicle {

```
    private int id;
    private String name;
    private String licenseNumber;
    public Vehicle(int id, String name, String licenseNumber) {
        this.id = id;
        this.name = name;
        this.licenseNumber = licenseNumber;
    }
}
class TwoWheeler extends Vehicle {
    private String steeringHandle;
    public TwoWheeler(int id, String name, String licenseNumber, String steeringHandle) {
```

```
        super(id, name, licenseNumber);
        this.steeringHandle = steeringHandle;
    }
}
class FourWheeler extends Vehicle {
    private String steeringWheel;
    ublic FourWheeler(int id, String name, String licenseNumber, String steeringWheel) {
        super(id, name, licenseNumber);
        this.steeringWheel = steeringWheel;
    }
}
```

```
Two-Wheeler:
ID: 1
Name: Bike
License Number: ABC123
Steering Handle: HandleBar

Four-Wheeler:
ID: 2
Name: Car
License Number: XYZ456
Steering Wheel: Steering Wheel
```

13. Write a program to implement the Multiple Inheritance (Bank Interface, Customer & Account classes).

Code:

```
interface Bank {
    void deposit(double amount);
    void withdraw(double amount);
}
class Customer implements Bank {
    private double balance;
    public Customer(double balance) {
        this.balance = balance;
    }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount);
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
            System.out.println("Insufficient balance");
        }
    }
    public void displayBalance() {
        System.out.println("Current Balance: $" + balance);
    }
}
class Account implements Bank {
    private double balance;
    public Account(double balance) {
        this.balance = balance;
    }
    public void deposit(double amount) {
```

```
    balance += amount;
        System.out.println("Deposited: $" + amount);
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
            System.out.println("Insufficient balance");
        }
    }
    public void displayBalance() {
        System.out.println("Current Balance: $" + balance);

    }
}
public class MultipleInheritanceBank {

    public static void main(String[] args) {
        Customer customer = new Customer(1000);
        customer.deposit(500);
        customer.withdraw(200);
        customer.displayBalance();
        Account account = new Account(2000);
        account.deposit(1000);
        account.withdraw(500);
        account.displayBalance();
    }
}
```

```
Deposited: $500.0
Withdrawn: $200.0
Current Balance: $1300.0
Deposited: $1000.0
Withdrawn: $500.0
Current Balance: $2500.0
```

14. Write a program to implement the Multiple Inheritance (Gross Interface, Employee & Salary classes).

Code:

```
interface Gross {
    double calculateGrossSalary(double basicSalary, double allowances);
}
class Employee implements Gross {
    public double calculateGrossSalary(double basicSalary, double allowances) {
        return basicSalary + allowances;
    }
}
class Salary implements Gross {
    public double calculateGrossSalary(double basicSalary, double allowances) {
        return basicSalary + allowances;
    }
}
public class MultipleInheritanceGross {

    public static void main(String[] args) {
        Employee employee = new Employee();
        double empGross = employee.calculateGrossSalary(50000, 10000);
```

```
        System.out.println("Employee Gross Salary: $" + empGross);
        Salary salary = new Salary();
        double salGross = salary.calculateGrossSalary(60000, 12000);
        System.out.println("Salary Gross Salary: $" + salGross);
    }
}
```

```
Employee Gross Salary: $60000.0
Salary Gross Salary: $72000.0
```

15. Program to create a interface 'Mango' and implement it in classes 'Winter' and 'Summer'.

Code:

```
interface Mango {
    void displaySeason();
}
class Winter implements Mango {
    public void displaySeason() {
        System.out.println("Winter mangoes are available from November to February.");
    }
}
class Summer implements Mango {
    public void displaySeason() {
        System.out.println("Summer mangoes are available from March to June.");
    }
}
public class MangoSeasons {
    public static void main(String[] args) {
        Winter winterMango = new Winter();
        System.out.print("Winter Mangoes: ");
        winterMango.displaySeason();
        Summer summerMango = new Summer();
        System.out.print("Summer Mangoes: ");

        summerMango.displaySeason();
    }
}
```

```
Winter Mangoes: Winter mangoes are available from November to February.
Summer Mangoes: Summer mangoes are available from March to June.
```

16. Program to implement the Multiple Inheritance (Exam Interface, Student & Result classes).

Code:

```
interface Exam {
    void displayExam();
}
class Student implements Exam {
    private String name;
    private int rollNumber;
    public Student(String name, int rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
    }
    public void displayExam() {
        System.out.println("Student Name: " + name);
        System.out.println("Roll Number: " + rollNumber);
    }
}
class Result implements Exam {
    private int marks;
    public Result(int marks) {
        this.marks = marks;
```

```
    }
    public void displayExam() {
        System.out.println("Marks Obtained: " + marks);
    }
}
public class MultipleInheritanceExam {
    public static void main(String[] args) {

        Student student = new Student("John", 101);
        Result result = new Result(85);
        System.out.println("Student Details:");
        student.displayExam();
        System.out.println("Exam Result:");
        result.displayExam();
    }
}
```

```
Student Details:
Student Name: John
Roll Number: 101
Exam Result:
Marks Obtained: 85
```

17. Program to demonstrate use of hierarchical inheritance using interface.

Code:

```
interface Shape {
    double calculateArea();
}
interface TwoDimensionalShape extends Shape {
    double calculatePerimeter();
}
interface ThreeDimensionalShape extends Shape {
    double calculateVolume();
}
class Circle implements TwoDimensionalShape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }

}
class Sphere implements ThreeDimensionalShape {
    private double radius;
    public Sphere(double radius) {
        this.radius = radius;
    }
    public double calculateArea() {
        return 4 * Math.PI * radius * radius;
    }
    public double calculateVolume() {
        return (4.0 / 3.0) * Math.PI * radius * radius * radius;
    }
}
```

```
public class HierarchicalInheritanceDemo {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());
        Sphere sphere = new Sphere(4);
        System.out.println("Sphere Area: " + sphere.calculateArea());
        System.out.println("Sphere Volume: " + sphere.calculateVolume());

    }
}
```

```
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Sphere Area: 201.06192982974676
Sphere Volume: 268.082573106329
```

18. Java program to Perform Payroll Using Interface (Multiple Inheritance).

Code:

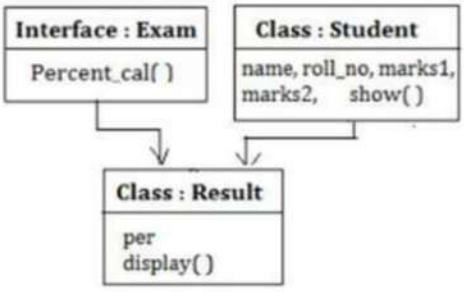
```
interface Payable {
    double calculatePay();
}
class Employee implements Payable {
    private String name;

    private double hourlyRate;
    private int hoursWorked;
    public Employee(String name, double hourlyRate, int hoursWorked) {
        this.name = name;
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }
    public double calculatePay() {
        return hourlyRate * hoursWorked;
    }
    public String getName() {
        return name;
    }
}
class Contractor implements Payable {
    private String name;
    private double rate;
    private int hoursWorked;
    public Contractor(String name, double rate, int hoursWorked) {
        this.name = name;
        this.rate = rate;
        this.hoursWorked = hoursWorked;
    }
    public double calculatePay() {
        return rate * hoursWorked;
    }
    public String getName() {
        return name;
    }
}
public class Payroll {
    public static void main(String[] args) {
        Employee employee = new Employee("John", 25.0, 40);
        Contractor contractor = new Contractor("Jane", 30.0, 30);
```

```
        displayPay(employee);
        displayPay(contractor);
    }
    public static void displayPay(Payable payable) {
        System.out.println("Name: " + payable.getName());
        System.out.println("Pay: $" + payable.calculatePay());
        System.out.println();
    }
}
```

```
Name: John
Pay: $1000.0

Name: Jane
Pay: $900.0
```

```
    private int marks1, marks2;
    public Student(String name, int roll_no, int marks1, int marks2) {
        this.name = name;
        this.roll_no = roll_no;
```

19. Implement the following diagram.



Code: interface Exam {

```
    double percent_calc();
}
class Student implements Exam {
    private String name;
    private int roll_no;
        this.marks1 = marks1;
        this.marks2 = marks2;

    }
    public double percent_calc() {
        int total_marks = marks1 + marks2;
```

```
    return (double) total_marks / 200 * 100;}
 public void show() {

}

class Result extends Student {

    private double percentage;

    public Result(String name, int roll_no, int marks1, int marks2) {

        super(name, roll_no, marks1, marks2);

        this.percentage = super.percent_calc();

    }

    public void display() {

        super.show();

        System.out.println("Percentage: " + percentage + "%");

    }

}
    System.out.println("Name: " + name);
    System.out.println("Roll No: " + roll_no);
    System.out.println("Marks 1: " + marks1);
    System.out.println("Marks 2: " + marks2);
}
```

```
Name: John Doe
Roll No: 1001
Marks 1: 85
Marks 2: 92
Percentage: 88.5%
```

Week: 7

1. **Write a Java program to show the use of all keywords for exception handling**

Code:

```
public class ExceptionHandlingKeywordsDemo {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
            int[] arr = new int[5];

            arr[10] = 50;
        } catch (ArithmeticException ae) {
            System.out.println("Arithmetic Exception occurred.");
        } catch (ArrayIndexOutOfBoundsException aioobe) {
            System.out.println("Array Index Out Of Bounds Exception occurred.");
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

```
Arithmetic Exception occurred.
Finally block executed.
```

2. Write a Java program using try and catch to generate NegativeArrayIndex Exception and Arithmetic Exception.

Code:

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

        try {
            int[] arr = new int[5];
            arr[-1] = 10;
            int result = 10 / 0;
        } catch (NegativeArraySizeException nae) {
            System.out.println("Negative Array Index Exception occurred.");
        } catch (ArithmeticException ae) {
            System.out.println("Arithmetic Exception occurred.");
        }
    }
}
```

3. Define an exception called "NoMatchFoundException" that is thrown when a string is not equal to "University". Write a program that uses this exception.

Code:

```
class NoMatchFoundException extends Exception {
    public NoMatchFoundException(String message) {
        super(message);
    }
}
public class NoMatchFoundDemo {

    public static void main(String[] args) {
        try {
            String inputString = "College";
            if (!inputString.equals("University")) {
                throw new NoMatchFoundException("Input string does not match 'University'");
            }
        } catch (NoMatchFoundException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

```
Input string does not match 'University'
```

4. Write a class that keeps a running total of all characters passed to it (one at a time) and throws anexception if it is passed a non-alphabetic character.

Code:

```
class NonAlphabeticCharacterException extends Exception {
    public NonAlphabeticCharacterException(String message) {
        super(message);
    }
}
public class CharacterTotal {
    private int total;
    public CharacterTotal() {
        total = 0;
    }
    public void addCharacter(char ch) throws NonAlphabeticCharacterException {
        if (!Character.isLetter(ch)) {
            throw new NonAlphabeticCharacterException("Non-alphabetic character
  encountered: " + ch);
        }
        total++;
    }
    public int getTotal() {
        return total;

    }
}
public class CharacterTotalDemo {
    public static void main(String[] args) {
        CharacterTotal characterTotal = new CharacterTotal();
        try {
            characterTotal.addCharacter('a');
            characterTotal.addCharacter('b');
            characterTotal.addCharacter('1');
            characterTotal.addCharacter('c');
        } catch (NonAlphabeticCharacterException e) {
            System.out.println(e.getMessage());
        }
        System.out.println("Total characters: " + characterTotal.getTotal());
    }
}
```

5. Write a program called Factorial.java that computes factorials and catches the result in an array of type long for reuse. The long type of variable has its own range. For example 20! Is as high as the range of long type. So check the argument passes and "throw an exception", if it is too big or too small.

- **If x is less than 0 throw an IllegalArgumentException with a message "Value of x must be positive".**
- **If x is above the length of the array throw an IllegalArgumentException with a message "Result will overflow". Here x is the value for which we want to find the factorial.**

Code:

```
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        try {
            long[] factorials = new long[n + 1];
            if (n < 0) {
                throw new IllegalArgumentException("Value of x must be positive");} else if (n >
                factorials.length - 1) {
                throw new IllegalArgumentException("Result will overflow");
            }
            factorials[0] = 1;
            for (int i = 1; i <= n; i++) {
            factorials[i] = factorials[i - 1] * i;
            }
            System.out.println("Factorial of " + n + " is: " + factorials[n]);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input format. Please provide a valid integer.");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
java Factorial 5
```

6. Write a class that keeps a running total of all characters passed to it (one at a time) and throws an exception if it is passed a non-alphabetic character.

Code:

```
class NonAlphabeticCharacterException extends Exception {
    public NonAlphabeticCharacterException(String message) {
        super(message);

    }
}
public class CharacterTotal {
    private int total;
    public CharacterTotal() {
        total = 0;
    }
    public void addCharacter(char ch) throws NonAlphabeticCharacterException {
        if (!Character.isLetter(ch)) {
            throw new NonAlphabeticCharacterException("Non-alphabetic character
encountered: " + ch);
        }
        total++;
    }
    public int getTotal() {
        return total;
    }
```

```
}
public class CharacterTotalDemo {
    public static void main(String[] args) {
        CharacterTotal characterTotal = new CharacterTotal();
        try {
            characterTotal.addCharacter('a');
            characterTotal.addCharacter('b');
            characterTotal.addCharacter('1'); // Non-alphabetic character
            characterTotal.addCharacter('c');
        } catch (NonAlphabeticCharacterException e) {
            System.out.println(e.getMessage());
        }
        System.out.println("Total characters: " + characterTotal.getTotal());
    }
}
```

```
Non-alphabetic character encountered: 1
Total characters: 2
```

7. Write a program that outputs the name of the capital of the country entered at the command line. The program should throw a "NoMatchFoundException" when it fails to print the capital of the country entered at the command line.

Code:

```
class NoMatchFoundException extends Exception {
    public NoMatchFoundException(String message) {
        super(message);
    }
}
public class CountryCapital {
    public static void main(String[] args) {
        try {
            String country = args[0].toLowerCase();
            String capital = getCapital(country);
            System.out.println("Capital of " + country + " is " + capital);
        } catch (NoMatchFoundException e) {
            System.out.println(e.getMessage());
        }
    }
    public static String getCapital(String country) throws NoMatchFoundException {
        switch (country) {

            case "india":

                return "New Delhi";
            case "usa":
                return "Washington D.C.";
            case "uk":
                return "London";
            default:
                throw new NoMatchFoundException("No capital found for country: " + country);
        }
    }
}
```

```
Capital of india is New Delhi
```

8. Write a program that takes a value at the command line for which factorial is to be computed. The program must convert the string to its integer equivalent. There are three possible user input errors that can prevent the program from executing normally.

- **The first error is when the user provides no argument while executing the program and an ArrayIndexOutOfBoundsException is raised. You must write a catch block for this.**
- **The second error is NumberFormatException that is raised in case the user provides a non-integer (float double) value at the command line.**
- **The third error is IllegalArgumentException. This needs to be thrown manually if the value at the command line is 0.**

Code:

```
public class FactorialCalculator {
    public static void main(String[] args) {
        try {
            if (args.length == 0) {
                throw new ArrayIndexOutOfBoundsException("No argument provided.");
            }
            int num = Integer.parseInt(args[0]);
            if (num <= 0) {
                throw new IllegalArgumentException("Value must be a positive integer greater
than 0.");
            }
            long factorial = calculateFactorial(num);
            System.out.println("Factorial of " + num + " is: " + factorial);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input format. Please provide a valid integer.");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
    public static long calculateFactorial(int n) {
        long factorial = 1;
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        return factorial;
    }
}
```

```
Factorial of 5 is: 120
```

9. Create a user-defined exception named CheckArgument to check the number of arguments passed through the command line. If the number of argument is less than 5, throw the CheckArgumentexception,else print the addition of all the five numbers.

Code:

```
class CheckArgumentException extends Exception {
    public CheckArgumentException(String message) {
        super(message);
    }
}
public class ArgumentChecker {
    public static void main(String[] args) {
        try {
            if (args.length < 5) {
                throw new CheckArgumentException("Number of arguments must be at least 5.");
```

```
            }
            int sum = 0;
            for (String arg : args) {
                sum += Integer.parseInt(arg);
            }

            System.out.println("Sum of all arguments: " + sum);
        } catch (CheckArgumentException e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input format. Please provide valid integers.");
        }
    }
}
```

```
Sum of all arguments: 15
```

10. Consider a Student examination database system that prints the mark sheet of students. Input the following from the command line.
(a) Student's Name
(b) Marks in six subjects
These marks should be between 0 to 50. If the marks are not in the specified range, raise a RangeException, else find the total marks and prints the percentage of the students.
Code:

```
class RangeException extends Exception { public RangeException(String message)
{
        super(message);
    }
}
public class MarkSheet {
    public static void main(String[] args) {
        try {
            if (args.length != 7) {
                throw new IllegalArgumentException("Invalid number of arguments. Expected 7.");
            }
            String name = args[0];
            int[] marks = new int[6];
            for (int i = 1; i < args.length; i++) {
                marks[i - 1] = Integer.parseInt(args[i]);
                if (marks[i - 1] < 0 || marks[i - 1] > 50) {
                    throw new RangeException("Marks should be between 0 to 50.");
                }
            }
            int totalMarks = 0;
            for (int mark : marks) {

            totalMarks += mark;
            }
            double percentage = (double) totalMarks / 300 * 100;
            System.out.println("Student Name: " + name);
            System.out.println("Total Marks: " + totalMarks);
            System.out.println("Percentage: " + percentage + "%");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Invalid input format. Please provide valid integers for marks.");
        } catch (RangeException e) {
            System.out.println(e.getMessage());
```

```
    }
    }
}
```

```
Student Name: John
Total Marks: 260
Percentage: 86.66666666666667%
```

11. Write a java program to create an custom Exception that would handle at least 2 kind
of Arithmetic Exceptions while calculating a given equation (e.g. X+Y*(P/Q)Z-I)
Code: // 11. Write a java program to create a custom Exception that would handle at least 2 kinds of Arithmetic Exceptions while calculating a given equation (e.g. X+Y*(P/Q)Z-I)

Code:
```
class CustomArithmeticException extends Exception {
    public CustomArithmeticException(String message) {
        super(message);
    }
}


public class EquationCalculator {
    public static void main(String[] args) {
        try {
            int x = Integer.parseInt(args[0]);
            int y = Integer.parseInt(args[1]);
            int p = Integer.parseInt(args[2]);
            int q = Integer.parseInt(args[3]);
            int z = Integer.parseInt(args[4]);
            int i = Integer.parseInt(args[5]);

            double result = calculateEquation(x, y, p, q, z, i);
            System.out.println("Result of the equation: " + result);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input format. Please provide valid integers.");
        } catch (CustomArithmeticException e) {
            System.out.println(e.getMessage());
        }
    }


    public static double calculateEquation(int x, int y, int p, int q, int z, int i) throws
CustomArithmeticException {
        try {
            return x + y * (p / q) * z - i;
        } catch (ArithmeticException ae) {
```

```
java EquationCalculator 10 5 20 0 3 2
```

```
    public TooCold(String message) {
        throw new CustomArithmeticException("Arithmetic Exception occurred while
calculating the equation.");
    }
  }
}
```

12. Create two user-defined exceptions named "TooHot" and "TooCold" to check the temperature (in Celsius) given by the user passed through the command line is too hot or too cold.

- **If temperature > 35, throw exception "TooHot".**
- **If temperature <5, throw exception "TooCold".**
- **Otherwise, print "Normal" and convert it to Farenheit.**

Code:

```
class TooHot extends Exception {
    public TooHot(String message) {
        super(message);
    }
}
class TooCold extends Exception {
        super(message);
    }
}
public class TemperatureCheck {
    public static void main(String[] args) {try {
            int temperature = Integer.parseInt(args[0]);
            if (temperature > 35) {
                throw new TooHot("Temperature is too hot!");
            }
            else if (temperature < 5) {
                throw new TooCold("Temperature is too cold!");
            }
            else {
                System.out.println("Normal");

                double fahrenheit = (temperature * 9.0 / 5.0) + 32.0;
                System.out.println("Temperature in Fahrenheit: " + fahrenheit);
            }
        } catch (TooHot e) {
            System.out.println(e.getMessage());
        } catch (TooCold e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Please provide a valid integer temperature as input.");
        }
```

```
        catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Please provide the temperature as a command line argument.");
        }
    }
}
```

```
Normal
Temperature in Fahrenheit: 77.0
```

13. Consider an Employee recruitment system that prints the candidate name based on the age criteria. The name and age of the candidate are taken as Input.Create two user-defined exceptions named "TooOlder" and "TooYounger"

- **If age>45, throw exception "TooOlder".**
- **If age<20, throw exception "TooYounger".**
- **Otherwise, print "Eligible" and print the name of the candidate.**

Code:

```
class TooOlder extends Exception {
    public TooOlder(String message) {
        super(message);
    }
}
class TooYounger extends Exception {
    public TooYounger(String message) {
        super(message);
    }
}
public class EmployeeRecruitment {
    public static void main(String[] args) {
        try {

            String name = args[0];
            int age = Integer.parseInt(args[1]);
            if (age > 45) {
                throw new TooOlder("Candidate is too old for recruitment!");
            } else if (age < 20) {
                throw new TooYounger("Candidate is too young for recruitment!");
            } else {
                System.out.println("Eligible");
                System.out.println("Candidate Name: " + name);
            }
        } catch (TooOlder e) {
            System.out.println(e.getMessage());
        } catch (TooYounger e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Please provide a valid integer age as input.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Please provide name and age as command line arguments.");
        }
    }
}
```

```
Eligible
Candidate Name: John
```

14. Consider a "Binary to Decimal" Number conversion system which only accepts binary number as Input. If user provides a decimal number a custom Exception "WrongNumberFormat" exception will be thrown. Otherwise, it will convert into decimal and print into the screen.

Code:

```
class WrongNumberFormat extends Exception {
    public WrongNumberFormat(String message) {
        super(message);
    }
}


public class BinaryToDecimalConverter {
    public static void main(String[] args) {
        try {
            String binaryNumber = args[0];
            if (!binaryNumber.matches("[01]+")) {
                throw new WrongNumberFormat("Input is not a binary number!");
            }
            int decimalNumber = Integer.parseInt(binaryNumber, 2);
            System.out.println("Decimal equivalent: " + decimalNumber);
        } catch (WrongNumberFormat e) {
            System.out.println(e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Please provide a valid binary number as input.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Please provide the binary number as a command line
argument.");
        }
    }
}
```

```
Decimal equivalent: 10
```

15. Write a Java Program that Implement the Nested Try Statements.

Code:

```
public class NestedTryExample {
    public static void main(String[] args) {
        try {
            System.out.println("Outer try block starts");
            int result = 10 / 0; // This will throw ArithmeticException
            try {
                System.out.println("Inner try block starts");
                String str = null;
                System.out.println(str.length());

                System.out.println("Inner try block ends");

            } catch (NullPointerException e) {

                System.out.println("Caught NullPointerException: " + e.getMessage());

            }

            System.out.println("Outer try block ends");

        } catch (ArithmeticException e) {

            System.out.println("Caught ArithmeticException: " + e.getMessage());

        }
    }
}
```

```
Outer try block starts
Caught ArithmeticException: / by zero
Outer try block ends
```

- **Java Program Which has a Class Called LessBalanceException Which returns the Statement that Says WithDraw Amount(_Rs) is Not Valid**
- **Java Program that has a Class Which Creates 2 Accounts, Both Account Deposit Money and One Account Tries to WithDraw more Money Which Generates a LessBalanceException Take Appropriate Action for the Same.**

Code:
```
class LessBalanceException extends Exception {
    public LessBalanceException(String message) {
        super(message);
    }
}
class Account {
    private double balance;
    private static final double MIN_BALANCE = 500; // Minimum balance required
    public Account() {
        balance = MIN_BALANCE; // Initialize balance with minimum balance
    }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Amount deposited: " + amount);
        System.out.println("Current balance: " + balance);
    }
    public void withdraw(double amount) throws LessBalanceException {
        if (balance - amount < MIN_BALANCE) {
            throw new LessBalanceException("Withdrawal amount exceeds available balance!");
        }
        balance -= amount;
        System.out.println("Amount withdrawn: " + amount);
        System.out.println("Current balance: " + balance);
    }
}

public class BankAccountManagement {

public static void main(String[] args) {
        try {
            Account account = new Account();
            account.deposit(1000);
            account.withdraw(700);
        } catch (LessBalanceException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
Amount deposited: 1000.0
Current balance: 1500.0
Amount withdrawn: 700.0
Current balance: 800.0
```

17. Consider a Library Management System, where a user wants to find a book. If the book is present in Library (Hint: Use predefined array), then it will print the book. Otherwise it will throw an exception "BookNotFoundException".

Code:
```
class BookNotFoundException extends Exception {
    public BookNotFoundException(String message) {
        super(message);
    }
}
public class LibraryManagementSystem {
    private static final String[] books = {"Book1", "Book2", "Book3", "Book4", "Book5"};
```

```
    public static void findBook(String bookTitle) throws BookNotFoundException {
        boolean found = false;
        for (String book : books) {
            if (book.equalsIgnoreCase(bookTitle)) {
                System.out.println("Book found: " + book);
                found = true;
                break;
            }
        }
        if (!found) {
             throw new BookNotFoundException("Book not found in the library!");
        }

    }
    public static void main(String[] args) {
        try {
            findBook("Book3");
            findBook("Book6");
        } catch (BookNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
Book found: Book3
Book not found in the library!
```

18. Consider a Quiz Management System, where a user needs to answer 5 questions. If any of the answer is wrong, throw an exception "NotCorrectException". If the answer is correct give a message "good! The answer is correct".

Code:

```
class NotCorrectException extends Exception {
    public NotCorrectException(String message) {
        super(message);
    }
}
public class QuizManagementSystem {
    public static void checkAnswer(String userAnswer, String correctAnswer, int
questionNumber) throws NotCorrectException {
        if (!userAnswer.equalsIgnoreCase(correctAnswer)) {
            throw new NotCorrectException("Incorrect answer for question " + questionNumber +
"!");
        }
        System.out.println("Good! The answer to question " + questionNumber + " is correct.");

    }
```

```
    public static void main(String[] args) {
        try {
            String[] questions = {"Q1: What is the capital of France?", "Q2: What is the capital of
Japan?",
                                  "Q3: What is the largest ocean?", "Q4: Who wrote Romeo and Juliet?",
                                  "Q5: What is the chemical symbol for water?"};

            String[] correctAnswers = {"Paris", "Tokyo", "Pacific", "William Shakespeare", "H2O"};
            String[] userAnswers = {"Paris", "Tokyo", "Atlantic", "William Shakespeare", "H2O"};
            for (int i = 0; i < questions.length; i++) {
                checkAnswer(userAnswers[i], correctAnswers[i], i + 1);

            }
        } catch (NotCorrectException e) {
            System.out.println(e.getMessage());
        }
    }
}
    public InvalidUsernameException(String message) {
        super(message);
    }
}
```

```
Good! The answer to question 1 is correct.
Good! The answer to question 2 is correct.
Incorrect answer for question 3!
```

19. Write a program to raise a user defined exception if username is less than 6 characters and password does not match.

Code: class InvalidsernameException extends Exception {

```
class PasswordMismatchException extends Exception {
    public PasswordMismatchException(String message) {
        super(message);
    }
}
public class AuthenticationSystem {
    public static void authenticate(String username, String password, String confirmPassword)
throws InvalidUsernameException, PasswordMismatchException {

        if (username.length() < 6) {
            throw new InvalidUsernameException("Username must be at least 6 characters
long!");
        }
        if (!password.equals(confirmPassword)) {
            throw new PasswordMismatchException("Passwords do not match!");
        }
        System.out.println("Authentication successful!");
    }
    public static void main(String[] args) {

        try {
            String username = "user123";
            String password = "password123";
            String confirmPassword = "password123"; // Correct password confirmation
            authenticate(username, password, confirmPassword);
        } catch (InvalidUsernameException | PasswordMismatchException e) {
            System.out.println(e.getMessage());
```

```
    }

  }
}
```

```
Authentication successful!
```

20. Write a program to accept a password from the user and throw 'Authentication Failure' exception if the password is incorrect.

Code:

```
class AuthenticationFailureException extends Exception {
  public AuthenticationFailureException(String message) {
    super(message);
  }
}
public class PasswordAuthentication {
  public static void authenticatePassword(String enteredPassword, String correctPassword)
throws AuthenticationFailureException {
    if (!enteredPassword.equals(correctPassword)) {
      throw new AuthenticationFailureException("Authentication failed! Incorrect
password.");
    }

    System.out.println("Authentication successful!");
  }
  public static void main(String[] args) {
    try {
      String correctPassword = "password123";
      String enteredPassword = "password123"; // Correct password
      authenticatePassword(enteredPassword, correctPassword);
    } catch (AuthenticationFailureException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

```
Authentication successful!
```

21. Write a program to input name and age of a person and throw a user-defined exception, if the entered age is negative.

Code:

```
class NegativeAgeException extends Exception {
  public NegativeAgeException(String message) {
    super(message);
  }
}
public class PersonInfo {
  public static void validateAge(int age) throws NegativeAgeException {
    if (age < 0) {
      throw new NegativeAgeException("Age cannot be negative!");
    }
    System.out.println("Name and age input successful!");
  }
  public static void main(String[] args) {
    try {
      String name = "John"; // Assume name is predefined
      int age = -25; // Negative age
      validateAge(age);
    } catch (NegativeAgeException e) {
      System.out.println(e.getMessage());
```

```
    }

  }
}
```

```
Age cannot be negative!
```

22. Write a program to throw user defined exception if the given number is not positive.

Code:

```
class NonPositiveNumberException extends Exception {
  public NonPositiveNumberException(String message) {
    super(message);
  }
}
public class PositiveNumberChecker {
  public static void checkPositiveNumber(int number) throws NonPositiveNumberException
{
    if (number <= 0) {
      throw new NonPositiveNumberException("Number must be positive!");
    }
    System.out.println("Number input successful!");
  }
  public static void main(String[] args) {
    try {
      int number = -10; // Non-positive number
      checkPositiveNumber(number);
    } catch (NonPositiveNumberException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

```
Number must be positive!
```

23. Write a program to throw a user-defined exception "String Mismatch Exception", if two strings are not equal (ignore the case).

Code:

```
class StringMismatchException extends Exception {
  public StringMismatchException(String message) {
    super(message);
  }
}
public class StringComparator {
  public static void compareStrings(String str1, String str2) throws StringMismatchException
{

    if (!str1.equalsIgnoreCase(str2)) {
      throw new StringMismatchException("Strings do not match!");
    }
    System.out.println("Strings match!");
  }
  public static void main(String[] args) {
    try {
      String str1 = "Hello";
      String str2 = "hello";
      compareStrings(str1, str2);
    } catch (StringMismatchException e) {
      System.out.println(e.getMessage());
    }
  }
```

```
}
```

```
Strings do not match!
```

24. Design a stack class. Provide your own stack exceptions namely push exception and pop exception, which throw exceptions when the stack is full and when the stack is empty respectively. Show the usage of these exceptions in handling a stack object in the main.

Code:

```
class PushException extends Exception {
    public PushException(String message) {
        super(message);
    }
}
class PopException extends Exception {
    public PopException(String message) {
        super(message);
    }
}
class MyStack {
    private int[] stackArray;
    private int top;

    private int maxSize;
    public MyStack(int size) {
        maxSize = size;
        stackArray = new int[maxSize];
        top = -1; // Empty stack initially
    }
    public void push(int element) throws PushException {
        if (top == maxSize - 1) {
            throw new PushException("Stack overflow! Cannot push element onto full stack.");
        }
        stackArray[++top] = element;
    }
    public int pop() throws PopException {
        if (top == -1) {
            throw new PopException("Stack underflow! Cannot pop element from empty stack.");
        }
        return stackArray[top--];
    }
    public boolean isEmpty() {
        return (top == -1);
    }
    public boolean isFull() {
        return (top == maxSize - 1);
    }
}
public class StackDemo {
    public static void main(String[] args) {
        MyStack stack = new MyStack(5);
        try {
            for (int i = 1; i <= 5; i++) {
                stack.push(i);
            }
            stack.push(6);
        } catch (PushException e) {
```

```
            System.out.println(e.getMessage());
        }
        try {
            while (!stack.isEmpty()) {
                System.out.println("Popped: " + stack.pop());
            }
            stack.pop();
        } catch (PopException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
Stack overflow! Cannot push element onto full stack.
Popped: 5
Popped: 4
Popped: 3
Popped: 2
Popped: 1
Stack underflow! Cannot pop element from empty stack.
```

25. Write an application that displays a series of at least five student ID numbers (that you have stored in an array) and asks the user to enter a numeric test score for the student. Create a ScoreException class, and throw a ScoreException for the class if the user does not enter a valid score (zero to 100). Catch the ScoreException and then display an appropriate message. In addition, store a 0 for the student's score. At the end of the application, display all the student IDs and scores.

Code:

```
class ScoreException extends Exception {
    public ScoreException(String message) {
        super(message);
    }
}
public class StudentScores {
    public static void main(String[] args) {
        int[] studentIDs = {101, 102, 103, 104, 105};
        int[] scores = new int[studentIDs.length];


        java.util.Scanner scanner = new java.util.Scanner(System.in);
        for (int i = 0; i < studentIDs.length; i++) {
            System.out.print("Enter test score for student " + studentIDs[i] + ": ");
            try {
                int score = scanner.nextInt();
                if (score < 0 || score > 100) {
                    throw new ScoreException("Invalid test score! Score must be between 0 and
100.");
                }
                scores[i] = score; // Store valid score
            } catch (ScoreException e)


                    System.out.println(e.getMessage());

                    scores[i] = 0;                                                    }

           System.out.println("\nStudent    IDs    and    Test
           Scores:"); for (int i = 0; i < studentIDs.length; i++) {
           System.out.println("Student ID: " + studentIDs[i] + ", Test Score: " + scores[i]);
```

}
scanner.close();

```
Enter test score for student 101: 90
Enter test score for student 102: 105
Invalid test score! Score must be between 0 and 100.
Enter test score for student 103: -5
Invalid test score! Score must be between 0 and 100.
Enter test score for student 104: 80
Enter test score for student 105: 95

Student IDs and Test Scores:
Student ID: 101, Test Score: 90
Student ID: 102, Test Score: 0
Student ID: 103, Test Score: 0
Student ID: 104, Test Score: 80
Student ID: 105, Test Score: 95
```

Week: 8

1. Write a Java program for calculating Factorial. Number should be taken through user input (Using Scanner, BufferedReader both).

Code:
```
import java.util.Scanner;
public class FactorialCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to calculate its factorial: ");
        int number = scanner.nextInt();
        int factorial = calculateFactorial(number);
        System.out.println("Factorial of " + number + " is: " + factorial);
        scanner.close();
    }
    private static int calculateFactorial(int n) {
        if (n == 0)
            return 1;
        else
            return n * calculateFactorial(n - 1);
    }
}
```

```
Enter a number to calculate its factorial: 8
Factorial of 8 is: 40320
```

2. Design a palindrome class that will input a string from console and check whether the string is palindrome or not.

Code:
```
import java.util.Scanner;

public class PalindromeChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string to check if it's a palindrome: ");
        String input = scanner.nextLine();
        if (isPalindrome(input))
            System.out.println(input + " is a palindrome.");
        else
            System.out.println(input + " is not a palindrome.");
        scanner.close();
    }
    private static boolean isPalindrome(String str) {
        StringBuilder reversed = new StringBuilder(str).reverse();
        return str.equals(reversed.toString());
    }
}
```

```
Enter a string to check if it's a palindrome: radar
radar is a palindrome.
```

3. Write a Java program to merge two strings.

Code:
```
import java.util.Scanner;
public class StringMerger {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");

        String second = scanner.nextLine();
```

```
        String merged = mergeStrings(first, second);
        System.out.println("Merged string: " + merged);
        scanner.close();
    }
    private static String mergeStrings(String first, String second) {
        return first + second;
    }
}
```

4. Write a Java program for reverse a string. (String will be taken as user input through console).

Code:

```
import java.util.Scanner;

public class StringReverser {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string to reverse: ");
        String input = scanner.nextLine();
        String reversed = reverseString(input);
        System.out.println("Reversed string: " + reversed);
        scanner.close();
    }
    private static String reverseString(String str) {
        return new StringBuilder(str).reverse().toString();
    }
}
```

```
Enter a string to reverse: UEM
Reversed string: MEU
```

5. Write a Java Program to Concatenate Two Strings.

Code:

```
import java.util.Scanner;

public class StringConcatenation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String second = scanner.nextLine();
        String concatenated = first.concat(second);
        System.out.println("Concatenated string: " + concatenated);
        scanner.close();
    }

}
```

```
Enter the first string: UEM
Enter the second string: K
Concatenated string: UEMK
```

6. Write a Java Program to check if a Given String is getChar from Specific Index.

Code:

```
import java.util.Scanner;
public class CharAtIndexChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");

        String str = scanner.nextLine();
```

```
        System.out.print("Enter an index: ");
        int index = scanner.nextInt();
        char character = getCharAtIndex(str, index);
        if (character != '\0')
            System.out.println("Character at index " + index + " is: " + character);
        else
            System.out.println("Invalid index.");
        scanner.close();
    }
    private static char getCharAtIndex(String str, int index) {
        if (index >= 0 && index < str.length())
            return str.charAt(index);
        else
            return '\0';
    }
}
```

```
Enter a string: UEMK
Enter an index: 2
Character at index 2 is: M
```

7. Write a Java Program to Find the Length of the String.

Code:

```
import java.util.Scanner;
public class StringLengthFinder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        int length = findStringLength(str);
        System.out.println("Length of the string is: " + length);
        scanner.close();
    }

    private static int findStringLength(String str) {
        return str.length();
    }
}
```

```
Enter a string: UEMK
Length of the string is: 4
```

8. Write a Java Program to Find All Possible Subsets of given Length in String.

Code :

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class SubsetsOfGivenLength {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        System.out.print("Enter the length of subsets: ");
        int length = scanner.nextInt();
        List<String> subsets = findAllSubsetsOfLength(str, length);
        System.out.println("All subsets of length " + length + " are:");
        for (String subset : subsets) {
            System.out.println(subset);
        }
```

```
        scanner.close();
    }

    private static List<String> findAllSubsetsOfLength(String str, int length) {
        List<String> subsets = new ArrayList<>();
        for (int i = 0; i <= str.length() - length; i++) {
            subsets.add(str.substring(i, i + length));
        }
        return subsets;
    }
}
```

```
Enter a string: UEMK
Enter the length of subsets: 2
All subsets of length 2 are:
UE
EM
MK
```

9. Write a Java Program to Remove the White Spaces from a String.

Code :

```
import java.util.Scanner;
public class RemoveWhiteSpace {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string with white spaces: ");
        String str = scanner.nextLine();
        String stringWithoutSpaces = removeWhiteSpace(str);
        System.out.println("String without white spaces: " + stringWithoutSpaces);
        scanner.close();
    }

    private static String removeWhiteSpace(String str) {
        return str.replaceAll("\\s", "");
    }
}
```

```
Enter a string with white spaces: u e m k
String without white spaces: uemk
```

10. Write a Java Program to Compare two Strings.

Code :

```
import java.util.Scanner;
public class StringComparer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String second = scanner.nextLine();

        boolean isEqual = compareStrings(first, second);
        if (isEqual)
            System.out.println("Both strings are equal.");
        else
            System.out.println("Strings are not equal.");
        scanner.close();

    }
```

```
    private static boolean compareStrings(String first, String second) {
        return first.equals(second);
    }
}
```

```
Enter the first string: UEM
Enter the second string: K
Strings are not equal.
```

11. Write a Java Program to Compare Performance of Two Strings.

Code :

```
public class StringPerformanceComparator {
    public static void main(String[] args) {
        String string1 = "Hello";
        String string2 = "World";
        long startTimeConcat1 = System.nanoTime();
        for (int i = 0; i < 100000; i++) {
            String result = string1 + string2;
        }
        long endTimeConcat1 = System.nanoTime();
        long durationConcat1 = endTimeConcat1 - startTimeConcat1;

        long startTimeBuilder1 = System.nanoTime();
        for (int i = 0; i < 100000; i++) {
            StringBuilder builder = new StringBuilder(string1);
            builder.append(string2);
            String result = builder.toString();
        }
        long endTimeBuilder1 = System.nanoTime();
        long durationBuilder1 = endTimeBuilder1 - startTimeBuilder1;
        long startTimeConcat2 = System.nanoTime();
        for (int i = 0; i < 100000; i++) {
            String result = string2 + string1;
        }
        long endTimeConcat2 = System.nanoTime();
        long durationConcat2 = endTimeConcat2 - startTimeConcat2;
        long startTimeBuilder2 = System.nanoTime();
        for (int i = 0; i < 100000; i++) {
            StringBuilder builder = new StringBuilder(string2);
            builder.append(string1);
            String result = builder.toString();
        }
        long endTimeBuilder2 = System.nanoTime();
        long durationBuilder2 = endTimeBuilder2 - startTimeBuilder2;
        System.out.println("Performance comparison of string concatenation:");
        System.out.println("String1 + String2: " + durationConcat1 + " nanoseconds");
        System.out.println("StringBuilder for String1: " + durationBuilder1 + " nanoseconds");
        System.out.println("String2 + String1: " + durationConcat2 + " nanoseconds");
        System.out.println("StringBuilder for String2: " + durationBuilder2 + " nanoseconds");

    }
}
```

```
Performance comparison of string concatenation:
String1 + String2: 16216000 nanoseconds
StringBuilder for String1: 18277900 nanoseconds
String2 + String1: 9156100 nanoseconds
StringBuilder for String2: 4513700 nanoseconds
```

12. Write a Java Program to Use Equals Method In a String Class.

Code :

```
import java.util.Scanner;
public class EqualsMethodUsage {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String second = scanner.nextLine();
        boolean areEqual = useEqualsMethod(first, second);
        if (areEqual)
            System.out.println("Both strings are equal.");
        else
            System.out.println("Strings are not equal.");
        scanner.close();
    }
    private static boolean useEqualsMethod(String first, String second) {
        return first.equals(second);
    }
}
```

```
Enter the first string: UEM
Enter the second string: Kolkata
Strings are not equal.
```

13. Write a Java Program to Use EqualsIgnoreCase Method In a String Class.

Code :

```
import java.util.Scanner;
public class EqualsIgnoreCaseMethodUsage {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String second = scanner.nextLine();
        boolean areEqualIgnoreCase = useEqualsIgnoreCaseMethod(first, second);
        if (areEqualIgnoreCase)
            System.out.println("Both strings are equal ignoring case.");
        else
            System.out.println("Strings are not equal ignoring case.");
        scanner.close();
    }
    private static boolean useEqualsIgnoreCaseMethod(String first, String second) {
        return first.equalsIgnoreCase(second);
    }
}
```

```
Enter the first string: UEM
Enter the second string: UEm
Both strings are equal ignoring case.
```

14. Write a Java Program to Use compareTo Method In a String Class.

Code :

```
import java.util.Scanner;
public class CompareToMethodUsage {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first string: ");
        String first = scanner.nextLine();
        System.out.print("Enter the second string: ");
        String second = scanner.nextLine();
```

```
        int comparisonResult = useCompareToMethod(first, second);
        if (comparisonResult == 0)
            System.out.println("Both strings are equal.");
        else if (comparisonResult < 0)
            System.out.println("First string is lexicographically smaller than the second string.");
        else
            System.out.println("First string is lexicographically greater than the second string.");
        scanner.close();
    }
    private static int useCompareToMethod(String first, String second) {
        return first.compareTo(second);
    }
}
```

```
Enter the first string: UEM
Enter the second string: UEM
Both strings are equal.
```

15. With a Java Program to Use compareToIgnoreCase Method In a String Class.

Code:

```
import java.util.Scanner;
public class CompareToIgnoreCase {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter first  string:");
        String str1 = scanner.nextLine();
        System.out.println("Enter second string:");
        String str2 = scanner.nextLine();
        int result = str1.compareToIgnoreCase(str2);
        if (result == 0) {
            System.out.println("Both strings are equal.");
        } else if (result < 0) {
            System.out.println("First string is lexicographically less than second string.");
        } else {
            System.out.println("First string is lexicographically greater than second string.");
        }
    }
}
```

```
Enter first string:
UEM
Enter second string:
Kolkata
First string is lexicographically greater than second string.
```

16. Write a Java Program to Replace Character or String.

Code:

```
import java.util.Scanner;
public class ReplaceCharacter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");

        String input = scanner.nextLine();
        System.out.println("Enter the character/string to replace:");
        String toReplace = scanner.nextLine();
        System.out.println("Enter the replacement character/string:");
        String replacement = scanner.nextLine();
        String replacedString = input.replace(toReplace, replacement);
        System.out.println("String after replacement:");
        System.out.println(replacedString);
    }
}
```

```
Enter a string:
UEMK
Enter the character/string to replace:
M
Enter the replacement character/string:
K
String after replacement:
UEKK
```

17. Write a Java Program to Search Last Occurance of a Substring Inside a Substring.

Code :
```
import java.util.Scanner;
public class LastOccurrence{
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter a string:");
      String input = scanner.nextLine();
      System.out.println("Enter the substring to search for:");
      String substring = scanner.nextLine();
      int lastIndex = input.lastIndexOf(substring);
      if (lastIndex != -1) {
         System.out.println("Last occurrence of substring is at index: " + lastIndex);
      } else {
         System.out.println("Substring not found in the string.");
      }
   }
}
```

```
Enter a string:
UEM
Enter the substring to search for:
E
Last occurrence of substring is at index: 1
```

18. Write a Java Program to Remove a Particular Character from a String.

Code:
```
import java.util.Scanner;
public class RemoveCharacter {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter a string:");
      String input = scanner.nextLine();
      System.out.println("Enter the character to remove:");
      char charToRemove = scanner.nextLine().charAt(0);
      String result = input.replaceAll(String.valueOf(charToRemove), "");


      System.out.println("String after removing '" + charToRemove + "':");
      System.out.println(result);
   }
}
```

```
Enter a string:
UEMK
Enter the character to remove:
K
String after removing 'K':
UEM
```

19. Write a Java Program to Replace a Substring Inside a String by Another One.

Code:
```
import java.util.Scanner;
public class ReplaceSubstring {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the substring to replace:");
        String toReplace = scanner.nextLine();
        System.out.println("Enter the replacement substring:");
        String replacement = scanner.nextLine();
        String replacedString = input.replace(toReplace, replacement);
        System.out.println("String after replacement:");
        System.out.println(replacedString);
    }
}
```

```
Enter a string:
UEMK
Enter the substring to replace:
E
Enter the replacement substring:
K
String after replacement:
UKMK
```

20. Write a Java Program to Reverse a String.

Code :

```
import java.util.Scanner;
public class ReverseString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        String reversed = reverseString(input);
        System.out.println("Reversed string:");
        System.out.println(reversed);
    }
    public static String reverseString(String str) {
        return new StringBuilder(str).reverse().toString();
    }
}
```

```
Enter a string:
madam
Reversed string:
madam
```

21. Write a Java Program to Search a Word Inside a String.

Code :

```
import java.util.Scanner;
public class WordSearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the word to search for:");
        String word = scanner.nextLine();
        boolean found = input.contains(word);
        if (found) {
            System.out.println("Word '" + word + "' found in the string.");
        } else {
            System.out.println("Word '" + word + "' not found in the string.");
        }
    }
```

```
}
```

```
Enter a string:
java
Enter the word to search for:
a
Word 'a' found in the string.
```

22. Write a Java Program to Split a String into a Number of Substrings.

Code :

```
import java.util.Scanner;
public class SplitString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the delimiter:");
        String delimiter = scanner.nextLine();
        String[] substrings = input.split(delimiter);
        System.out.println("Substrings:");
        for (String substring : substrings) {
            System.out.println(substring);
        }
    }
}
```

```
Enter a string:
UEM,UEMK
Enter the delimiter:
,
Substrings:
UEM
UEMK
```

23. Write a Java Program to Search a Particular Word in a String.

```
import java.util.Scanner;

public class WordSearch {
    public static void main(String[] args) {  Scanner
        scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String inputString = scanner.nextLine();

        // Word to search
        System.out.println("Enter the word to search:");
        String searchWord = scanner.next();
        String[] words = inputString.split("\\s+");
        boolean found = false;
        for (String word : words) {
            if (word.equalsIgnoreCase(searchWord)) { found =
                true;
                break;
            }
        }
        if (found) {
            System.out.println("The word '" + searchWord + "' is found in the string.");
        } else {
            System.out.println("The word '" + searchWord + "' is not found in the string.");
        }

        scanner.close();

    }
```

```
Enter a string:
The quick brown fox jump over the lazy dog
Enter the word to search:
quick
The word 'quick' is found in the string.
```

24. Write a Java Program to Replace All Occurings of a String.

Code:

```
import java.util.Scanner;

public class ReplaceOccurrences {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the string to replace:");
        String toReplace = scanner.nextLine();
        System.out.println("Enter the replacement string:");
        String replacement = scanner.nextLine();

        String replacedString = input.replaceAll(toReplace, replacement);

        System.out.println("Replaced string:");
        System.out.println(replacedString);
    }
}
```

```
Enter a string:
test
Enter the string to replace:
t
Enter the replacement string:
java
Replaced string:
javaesjava
```

25. Write a Java Program to Make First Character of Each Word in Uppercase.

Code :

```
import java.util.Scanner;
public class UppercaseFirstLetter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.println("Enter a string:");
        String input = scanner.nextLine();

        String[] words = input.split("\\s+");
        StringBuilder result = new StringBuilder();
        for (String word : words) {
            if (!word.isEmpty()) {
                result.append(Character.toUpperCase(word.charAt(0)))
                      .append(word.substring(1)).append(" ");
            }
        }
        System.out.println("String with first letter of each word in uppercase:");
        System.out.println(result.toString().trim());
    }
}
```

```
Enter a string:
java language
String with first letter of each word in uppercase:
Java Language
```

26. Write a Java Program to Delete All Repeated Words in String.

Code :

```
import java.util.Scanner;
public class RemoveChar {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String first = scanner.nextLine();
        System.out.println("Enter the second string:");
        String second = scanner.nextLine();

        StringBuilder result = new StringBuilder();
        for (int i = 0; i < second.length(); i++) {
            char currentChar = second.charAt(i);
            if (first.indexOf(currentChar) == -1) {
                result.append(currentChar);
            }
        }

        System.out.println("Resultant string after removal:");
        System.out.println(result.toString());
    }
}
```

27. Write a Java Program to Reverse the String Using Both Recursion and Iteration.

Code :

```
import java.util.Scanner;
public class ReverseString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        String reversed = reverseString(input);
        System.out.println("Reversed string:");
        System.out.println(reversed);
    }

    public static String reverseString(String str) {
        return new StringBuilder(str).reverse().toString();
    }
}
```

```
Enter a string:
kolkata
Reversed string:
ataklok
```

28. Write a Java Program to Convert a String Totally into Upper Case.

Code :

```
import java.util.Scanner;
public class UpperCaseConversion {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        String upperCase = input.toUpperCase();
        System.out.println("String in upper case:");
        System.out.println(upperCase);
    }
}
```

```
Enter a string:
kolkata
String in upper case:
KOLKATA
```

29. Write a Java Program to Remove all Characters in Second String which are Present in First String.

Code :

```
import java.util.Scanner;
public class ReplaceCharacter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the character/string to replace:");
        String toReplace = scanner.nextLine();
        System.out.println("Enter the replacement character/string:");
        String replacement = scanner.nextLine();
        String replacedString = input.replace(toReplace, replacement);
        System.out.println("String after replacement:");
        System.out.println(replacedString);
    }
}
```

```
Enter a string:
java
Enter the character/string to replace:
a
Enter the replacement character/string:
aa
String after replacement:
jaavaa
```

30. Write a Java Program to Find the Consecutive Occurrence of any Vowel in a String.

Code :

```
import java.util.Scanner;
public class VowelConsecutiveOccurrences {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        boolean hasConsecutiveVowels = checkConsecutiveVowels(input);
        if (hasConsecutiveVowels) {
            System.out.println("Consecutive occurrence of vowels found.");
        } else {
            System.out.println("No consecutive occurrence of vowels found.");
        }
    }

    public static boolean checkConsecutiveVowels(String input) {
        for (int i = 0; i < input.length() - 1; i++) {
            char current = input.charAt(i);
            char next = input.charAt(i + 1);
            if (isVowel(current) && isVowel(next)) {
                return true;
            }
        }
        return false;
    }

    public static boolean isVowel(char c) {
```

```
        return "AEIOUaeiou".indexOf(c) != -1;
        }
    }
```

```
Enter a string:
jaavaa
Consecutive occurrence of vowels found.
```

31. Write a Java Program to Find the Largest & Smallest Word in a String.

Code :

```
import java.util.Scanner;
public class LargestSmallestWord {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();

        String[] words = input.split("\\s+");
        String smallest  =  words[0];
        String largest = words[0];
        for (String word : words) {
            if (word.length() < smallest.length()) {
                smallest = word;
            }
            if (word.length() > largest.length()) {
                largest = word;
            }
        }


        System.out.println("Smallest word: " + smallest);
        System.out.println("Largest word: " + largest);
    }
}
```

```
Enter a string:
java is language.
Smallest word: is
Largest word: language.
```

32. Write a Java Program to Find First and Last Occurrence of Given Character in a String.

Code :

```
import java.util.Scanner;
public class FirstLastOccurrence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the character to search for:");
        char ch = scanner.next().charAt(0);
        int firstIndex = input.indexOf(ch);
        int lastIndex = input.lastIndexOf(ch);

        if (firstIndex == -1) {
            System.out.println("Character '" + ch + "' not found in the string.");
        } else {
            System.out.println("First occurrence of '" + ch + "' at index: " + firstIndex);
            System.out.println("Last occurrence of '" + ch + "' at index: " + lastIndex);
        }
    }
}
```

```
Enter a string:
kolkata
Enter the character to search for:
k
First occurrence of 'k' at index: 0
Last occurrence of 'k' at index: 3_
```

33. Write a Java Program to Display the Characters in Prime Position a Given String.

Code :

```
import java.util.Scanner;

public class PrimePositionCharacters {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Characters in prime positions:");
        for (int i = 2; i < input.length(); i++) {
            if (isPrime(i)) {
                System.out.print(input.charAt(i) + " ");
            }
        }
    }

    public static boolean isPrime(int n) {
        if (n <= 1) {
            return false;
        }
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

```
Enter a string:
kolkata
Characters in prime positions:
l k t
```

34. Write a Java Program to Sort String Ignoring Whitespaces and Repeating Characters Only Once.

Code :

```
import java.util.Arrays;
import java.util.Scanner;
public class SortStringIgnoringSpaces {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        input = input.replaceAll("\\s", "");
        char[] chars = input.toCharArray();
        Arrays.sort(chars);

        StringBuilder result = new StringBuilder();
        result.append(chars[0]);
        for (int i = 1; i < chars.length; i++) {
            if (chars[i] != chars[i - 1]) {
                result.append(chars[i]);
            }
        }
```

```
        System.out.println("Sorted string ignoring whitespaces and repeating characters once:");
        System.out.println(result.toString());
    }
}
```

```
Enter a string:
UEM Kolkata
Sorted string ignoring whitespaces and repeating characters once:
EKMUaklot
```

35. Write a Java Program to Count Replace First Occurrence of a String.

Code :

```
import java.util.Scanner;
public class CountReplaceFirstOccurrence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the input string:");
        String input = scanner.nextLine();
        System.out.println("Enter the string to replace:");
        String toReplace = scanner.nextLine();
        System.out.println("Enter the replacement string:");

        String replacement = scanner.nextLine();

        int count = 0;
        int index = input.indexOf(toReplace);
        if (index != -1) {
            count++;
            input = input.substring(0, index) + replacement + input.substring(index +
toReplace.length());
        }

        System.out.println("String after replacing first occurrence:");
        System.out.println(input);
        System.out.println("Number of replacements made: " + count);
    }
}
```

```
Enter the input string:
JAVA
Enter the string to replace:
AB
Enter the replacement string:
A
String after replacing first occurrence:
JAVA
Number of replacements made: 0
```

36. Write a Java Program to Know the Last Index of a Particular Word in a String.

Code :

```
import java.util.Scanner;
public class LastIndexOfWord {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the word to search for:");
        String word = scanner.nextLine();
        int lastIndex = input.lastIndexOf(word);
        if (lastIndex != -1) {
            System.out.println("Last index of '" + word + "' in the string: " + lastIndex);
        } else {
            System.out.println("Word '" + word + "' not found in the string.");
        }
    }
}
```

```
Enter a string:
Kolkata
Enter the word to search for:
a
Last index of 'a' in the string: 6
```

37. Write a Java Program to Access the Index of the Character or String.

Code :

```
import java.util.Scanner;

public class IndexOfCharacterString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");

        String input = scanner.nextLine();
        System.out.println("Enter the character/string to search for:");
        String search = scanner.nextLine();
        int index = input.indexOf(search);
        if (index != -1) {
            System.out.println("Index of '" + search + "' in the string: " + index);
        } else {
            System.out.println("'" + search + "' not found in the string.");
        }
    }
}
```

```
Enter a string:
KOLKATA
Enter the character/string to search for:
A
Index of 'A' in the string: 4
```

38. Write a Java Program to Access the Characters or the ASCII of the Character Available in the String.

Code

```
import java.util.Scanner;
public class CharactersASCIIInString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();

        System.out.println("Characters and corresponding ASCII values:");
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            int asciiValue = (int) c;
            System.out.println("'" + c + "': " + asciiValue);
        }
    }
}
```

```
Enter a string:
Kolkata
Characters and corresponding ASCII values:
'K': 75
'o': 111
'l': 108
'k': 107
'a': 97
't': 116
'a': 97
```

39. Write a Java Program to Display the Character and the Corresponding Ascii Present in the String.

Code :
```
import java.util.Scanner;
public class CharacterASCII {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a  character:");
        char c = scanner.next().charAt(0);

        int asciiValue = (int) c;
        System.out.println("ASCII value of '" + c + "': " + asciiValue);
    }
}
```

```
Enter a character:
A
ASCII value of 'A': 65
```

40. Write a Java Program to Accept 2 String & Check Whether all Characters in First String is Present in Second String & Print.

Code :
```
import java.util.Scanner;

public class CheckCharactersInSecondString {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String first = scanner.nextLine();
        System.out.println("Enter the second string:");
        String second = scanner.nextLine();

        boolean allPresent = true;
        for (char c : first.toCharArray()) {
            if (second.indexOf(c) == -1) {
                allPresent = false;
                break;
            }
        }

        if (allPresent) {
            System.out.println("All characters in the first string are present in the second string.");
        } else {
            System.out.println("Not all characters in the first string are present in the second
string.");
        }
    }
}
```

```
Enter the first string:
java
Enter the second string:
jaava
All characters in the first string are present in the second string.
```

41. Write a Java Program to Check whether a Given Character is Present in a String, Find Frequency & Position of Occurrence.

Code :
```
import java.util.Scanner;

public class CharacterOccurrence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the character to search for:");
        char target = scanner.next().charAt(0);
```

```
        int frequency = 0;
        for (int i = 0; i < input.length(); i++) {
            if (input.charAt(i) == target) {
                frequency++;
                System.out.println("Character '" + target + "' found at position: " + i);
            }
        }
        if (frequency == 0) {
            System.out.println("Character '" + target + "' not found in the string.");
        } else {
            System.out.println("Frequency of character '" + target + "': " + frequency);
        }
    }
}
```

```
Enter a string:
KOLKATA
Enter the character to search for:
K
Character 'K' found at position: 0
Character 'K' found at position: 3
Frequency of character 'K': 2
```

42. Write a Java Program to Count the Number of Occurrence of Each Character Ignoring the Case of Alphabets & Display them.

Code :

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class CharacterFrequencyIgnoringCase {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine().toLowerCase();
        Map<Character, Integer> frequencyMap = new HashMap<>();
        for (char c : input.toCharArray()) {
            if (Character.isAlphabetic(c)) {
                frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
            }
        }

        System.out.println("Character frequencies (ignoring case):");
        for (Map.Entry<Character, Integer> entry : frequencyMap.entrySet()) {
            System.out.println("'" + entry.getKey() + "': " + entry.getValue());
        }
    }
}
```

```
Enter a string:
java
Character frequencies (ignoring case):
'a': 2
'v': 1
'j': 1
```

43. Write a Java Program to Give Shortest Sequence of Character Insertions and Deletions that Turn One String Into the Other.

Code :

```
import java.util.Scanner;

public class ShortestSequence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String str1 = scanner.nextLine();
        System.out.println("Enter the second string:");
        String str2 = scanner.nextLine();

        int[][] dp = new int[str1.length() + 1][str2.length() + 1];
        for (int i = 0; i <= str1.length(); i++) {
            for (int j = 0; j <= str2.length(); j++) {
                if (i == 0 || j == 0)
                    dp[i][j] = i + j;
                else if (str1.charAt(i - 1) == str2.charAt(j - 1))
                    dp[i][j] = dp[i - 1][j - 1];
                else
                    dp[i][j] = 1 + Math.min(dp[i - 1][j], dp[i][j - 1]);
            }
        }

        int i = str1.length();
        int j = str2.length();
        StringBuilder sequence = new StringBuilder();
        while (i > 0 && j > 0) {
            if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
                i--;
                j--;
            } else if (dp[i - 1][j] < dp[i][j - 1]) {
                sequence.append("Delete ").append(str1.charAt(i - 1)).append(", ");
                i--;
            } else {
                sequence.append("Insert ").append(str2.charAt(j - 1)).append(", ");
                j--;
            }
        }
        while (i > 0) {
            sequence.append("Delete ").append(str1.charAt(i - 1)).append(", ");
            i--;
        }
        while (j > 0) {
            sequence.append("Insert ").append(str2.charAt(j - 1)).append(", ");
            j--;
        }

        if (sequence.length() > 0) {
            System.out.println("Shortest sequence of insertions and deletions:");
            System.out.println(sequence.substring(0, sequence.length() - 2));
        } else {
            System.out.println("No sequence of insertions and deletions needed.");
        }
    }
```

```
}
Enter the first string:
UEM
Enter the second string:
UEMK
Shortest sequence of insertions and deletions:
Insert K
```

44. Write a Java Program to Check Whether Date is in Proper Format or Not.

Code :
```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Scanner;
public class DateValidation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a date (in format dd-MM-yyyy):");
        String dateStr = scanner.nextLine();
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
        dateFormat.setLenient(false);
        try {
            dateFormat.parse(dateStr);
            System.out.println("Date is in proper format.");
        } catch (ParseException e) {
            System.out.println("Date is not in proper format.");
        }
    }
}
```

```
Enter a date (in format dd-MM-yyyy):
18-03-2024
Date is in proper format.
```

45. Write a Java Program to Validate an Email Address Format.

Code :
```
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class EmailValidation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter an email address:");
        String email = scanner.nextLine();
        String regex = "^[a-zA-Z0-9_+&*-]+(?:\\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-
Z]{2,7}$";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(email);

        if (matcher.matches()) {
            System.out.println("Email address is valid.");
        } else {
            System.out.println("Email address is not valid.");
        }
    }
}
```

```
C:\Users\DELL\Desktop\Program>javac EmailValidation.java

C:\Users\DELL\Desktop\Program>java EmailValidation
Enter an email address:
uem20@gmail.com
Email address is valid.
```

46. Write a Java Program to Store String Literals Using String Buffer.

Code :
```
public class StringBufferExample {
  public static void main(String[] args) {
     StringBuffer buffer = new StringBuffer();
     buffer.append("Hello");
     buffer.append(" ");
     buffer.append("World");

     System.out.println(buffer.toString());
  }
}
```

```
Hello World
```

47. Write a Java Program to Verify a Class is StringBuffer Class Method.

Code :
```
public class StringBufferCheck {
  public static void main(String[] args) {
     String str = "Hello";
     boolean isStringBuffer = str.getClass().equals(StringBuffer.class);
     System.out.println("Is 'str' an instance of StringBuffer? " + isStringBuffer);
  }
}
```

```
Is 'str' an instance of StringBuffer? false
```

48. Write a Java Program to Ask the User His Name and Greets Him With His Name.

Code :
```
import java.util.Scanner;
public class GreetWithName {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter your name:");
      String name = scanner.nextLine();

      System.out.println("Hello, " + name + "! Nice to meet you.");
   }
}
```

```
Enter your name:
Debagni Bhattacharjee
Hello, Debagni Bhattacharjee! Nice to meet you.
PS C:\Users\User\Desktop\Java\practice>
```

49. Code :
```
import java.util.Scanner;
public class WordGroupCount {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        System.out.println("Enter the word or group of words to count:");
        String wordGroup = scanner.nextLine();
        int count = countWordGroupOccurrences(input, wordGroup);
        System.out.println("Number of occurrences of the word/group: " + count);
    }

    public static int countWordGroupOccurrences(String input, String wordGroup) {
        int count = 0;
        int index = input.indexOf(wordGroup);
        while (index != -1) {
            count++;
            index = input.indexOf(wordGroup, index + 1);
        }
        return count;
    }

}
```

```
Enter a string:
Iam David,a resident of Kolkata
Enter the word or group of words to count:
a
Number of occurrences of the word/group: 5
PS C:\Users\User\Desktop\Java\practice>
```

50. Write a Java Program to Count Number of Words in a given Text or Sentence.

Code :

```
import java.util.Scanner;
public class WordCount {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a text or sentence:");
        String text = scanner.nextLine();

        int wordCount = countWords(text);
        System.out.println("Number of words: " + wordCount);
    }
    public static int countWords(String text) {
        String[] words = text.split("\\s+");
        return words.length;
    }
}
```

```
Enter a text or sentence:
Indus Valley
Number of words: 2
PS C:\Users\User\Desktop\Java\practice>
```