WEEK – 6

**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.**

import java.util.Scanner;

abstract class Shape {

abstract double calculateArea();

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;

}

@Override

double calculateArea() {

return length \* width;

}

@Override

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;

}

@Override

double calculateArea() {

double s = (side1 + side2 + side3) / 2;

return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

@Override

double calculatePerimeter() {

return side1 + side2 + side3;

}

}

public class q1\_shape\_triangle\_rectangle {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the dimensions of the rectangle (length and width):");

double length = sc.nextDouble();

double width = sc.nextDouble();

Rectangle r = new Rectangle(length, width);

System.out.println("Area of the rectangle: " + r.calculateArea());

System.out.println("Perimeter of the rectangle: " + r.calculatePerimeter());

System.out.println("Enter the dimensions of the triangle (side1, side2, side3):");

double side1 = sc.nextDouble();

double side2 = sc.nextDouble();

double side3 = sc.nextDouble();

Triangle t = new Triangle(side1, side2, side3);

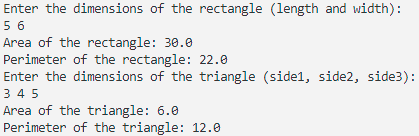
System.out.println("Area of the triangle: " + t.calculateArea());

System.out.println("Perimeter of the triangle: " + t.calculatePerimeter());

sc.close();

}

}



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

interface Animal {

void makeSound();

}

class Dog implements Animal {

@Override

public void makeSound() {

System.out.println("Dog barks: Woof! Woof!");

}

}

class Cat implements Animal {

@Override

public void makeSound() {

System.out.println("Cat meows: Meow! Meow!");

}

}

public class q2\_interface\_eg {

public static void main(String[] args) {

Dog dog = new Dog();

Cat cat = new Cat();

dog.makeSound();

cat.makeSound();

}

}



**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.**

import java.util.Scanner;

class ThreeDObject {

public double wholeSurfaceArea() {

return 0;

}

public double volume() {

return 0;

}

}

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;

}

@Override

public double wholeSurfaceArea() {

return 2 \* ((length \* width) + (length \* height) + (width \* height));

}

@Override

public double volume() {

return length \* width \* height;

}

}

class Cube extends ThreeDObject {

private double side;

public Cube(double side) {

this.side = side;

}

@Override

public double wholeSurfaceArea() {

return 6 \* side \* side;

}

@Override

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;

}

@Override

public double wholeSurfaceArea() {

return 2 \* Math.PI \* radius \* (radius + height);

}

@Override

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;

}

@Override

public double wholeSurfaceArea() {

double slantHeight = Math.sqrt(radius \* radius + height \* height);

return Math.PI \* radius \* (radius + slantHeight);

}

@Override

public double volume() {

return (1.0 / 3.0) \* Math.PI \* radius \* radius \* height;

}

}

public class q3\_three\_object {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the dimensions of the box (length, width, height):");

double length = sc.nextDouble();

double width = sc.nextDouble();

double height = sc.nextDouble();

Box box = new Box(length, width, height);

System.out.println("Volume of the box: " + box.volume());

System.out.println("Whole surface area of the box: " + box.wholeSurfaceArea());

System.out.println("\nEnter the side of the cube:");

double side = sc.nextDouble();

Cube cube = new Cube(side);

System.out.println("Volume of the cube: " + cube.volume());

System.out.println("Whole surface area of the cube: " + cube.wholeSurfaceArea());

System.out.println("\nEnter the dimensions of the cylinder (radius, height):");

double radius = sc.nextDouble();

height = sc.nextDouble();

Cylinder cylinder = new Cylinder(radius, height);

System.out.println("Volume of the cylinder: " + cylinder.volume());

System.out.println("Whole surface area of the cylinder: " +

cylinder.wholeSurfaceArea());

System.out.println("\nEnter the dimensions of the cone (radius, height):");

radius = sc.nextDouble();

height = sc.nextDouble();

Cone cone = new Cone(radius, height);

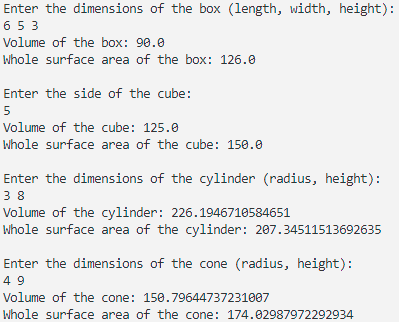
System.out.println("Volume of the cone: " + cone.volume());

System.out.println("Whole surface area of the cone: " + cone.wholeSurfaceArea());

sc.close();

}

}



**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.**

import java.util.Scanner;

class Vehicle {

protected String regnNumber;

protected int speed;

protected String color;

protected String ownerName;

public Vehicle(String regnNumber, int 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");

System.out.println("Registration Number: " + regnNumber);

System.out.println("Speed: " + speed);

System.out.println("Color: " + color);

System.out.println("Owner Name: " + ownerName);

}

}

class Bus extends Vehicle {

private String routeNumber;

public Bus(String regnNumber, int speed, String color, String ownerName, String routeNumber) {

super(regnNumber, speed, color, ownerName);

this.routeNumber = routeNumber;

}

@Override

protected void showData() {

super.showData();

System.out.println("Route Number: " + routeNumber);

}

}

class Car extends Vehicle {

private String manufacturerName;

public Car(String regnNumber, int speed, String color, String ownerName, String manufacturerName) {

super(regnNumber, speed, color, ownerName);

this.manufacturerName = manufacturerName;

}

@Override

protected void showData() {

super.showData();

System.out.println("Manufacturer Name: " + manufacturerName);

}

}

public class q4\_vehicle\_bus\_car {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter vehicle details:");

System.out.print("Registration Number: ");

String regnNumber = sc.nextLine();

System.out.print("Speed: ");

int speed = sc.nextInt();

sc.nextLine(); // Consume newline

System.out.print("Color: ");

String color = sc.nextLine();

System.out.print("Owner Name: ");

String ownerName = sc.nextLine();

System.out.println("Enter Bus details:");

System.out.print("Route Number: ");

String routeNumber = sc.nextLine();

Bus bus = new Bus(regnNumber, speed, color, ownerName, routeNumber);

System.out.println("\nEnter Car details:");

System.out.print("Manufacturer Name: ");

String manufacturerName = sc.nextLine();

Car car = new Car(regnNumber, speed, color, ownerName, manufacturerName);

System.out.println("\nBus Details:");

bus.showData();

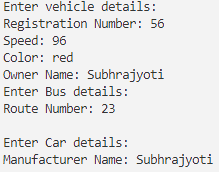
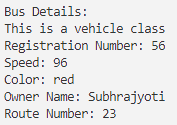
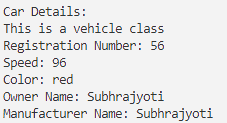
System.out.println("\nCar Details:");

car.showData();

sc.close();

}

}

**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.**

interface Interface1 {

void method1a();

void method1b();

}

interface Interface2 {

void method2a();

void method2b();

}

interface Interface3 {

void method3a();

void method3b();

}

interface CombinedInterface extends Interface1, Interface2, Interface3 {

void newMethod();

}

class MyClass extends SomeConcreteClass implements CombinedInterface {

@Override

public void method1a() {

System.out.println("Implementation of method1a");

}

@Override

public void method1b() {

System.out.println("Implementation of method1b");

}

@Override

public void method2a() {

System.out.println("Implementation of method2a");

}

@Override

public void method2b() {

System.out.println("Implementation of method2b");

}

@Override

public void method3a() {

System.out.println("Implementation of method3a");

}

@Override

public void method3b() {

System.out.println("Implementation of method3b");

}

@Override

public void newMethod() {

System.out.println("Implementation of newMethod");

}

}

class SomeConcreteClass {

}

public class q5\_interface4s {

public static void methodWithInterface1(Interface1 obj) {

System.out.println("Method with Interface1 argument");

obj.method1a();

obj.method1b();

}

public static void methodWithInterface2(Interface2 obj) {

System.out.println("Method with Interface2 argument");

obj.method2a();

obj.method2b();

}

public static void methodWithInterface3(Interface3 obj) {

System.out.println("Method with Interface3 argument");

obj.method3a();

obj.method3b();

}

public static void methodWithCombinedInterface(CombinedInterface obj) {

System.out.println("Method with CombinedInterface argument");

obj.method1a();

obj.method1b();

obj.method2a();

obj.method2b();

obj.method3a();

obj.method3b();

obj.newMethod();

}

public static void main(String[] args) {

MyClass myObj = new MyClass();

methodWithInterface1(myObj);

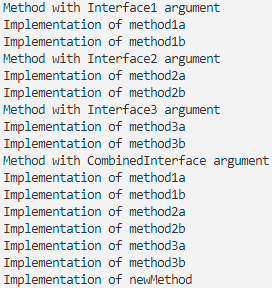
methodWithInterface2(myObj);

methodWithInterface3(myObj);

methodWithCombinedInterface(myObj);

}

}



**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.**

import java.util.\*;

interface department {

String deptName = "";

String deptHead = "";

abstract void Department();

}

class hostel {

String hostelName, hostelLocation;

int numberOfRooms;

public void setHostelName(String hostelName) {

this.hostelName = hostelName;

}

public void setHostelLocation(String hostelLocation) {

this.hostelLocation = hostelLocation;

}

public void setNumberOfRooms(int numberOfRooms) {

this.numberOfRooms = numberOfRooms;

}

public String getHostelName() {

return hostelName;

}

public String getHostelLocation() {

return hostelLocation;

}

public int getNumberOfRooms() {

return numberOfRooms;

}

void Hostel() {

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 {

String studentName, electiveSubject, deptName, deptHead, hostelName, hostelLocation;

int regdNo, numberOfRooms;

double avgMarks;

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public void setElectiveSubject(String electiveSubject) {

this.electiveSubject = electiveSubject;

}

public void setRegdNo(int regdNo) {

this.regdNo = regdNo;

}

public void setAvgMarks(double avgMarks) {

this.avgMarks = avgMarks;

}

public void setDeptName(String deptName) {

this.deptName = deptName;

}

public void setDeptHead(String deptHead) {

this.deptHead = deptHead;

}

public String getStudentName() {

return studentName;

}

public String getElectiveSubject() {

return electiveSubject;

}

public int getregdNo() {

return regdNo;

}

public double getAvgMarks() {

return avgMarks;

}

public String getDeptName() {

return deptName;

}

public String getDeptHead() {

return deptHead;

}

public void setData(String studentName, String electiveSubject, int regdNo, double avgMarks, String deptHead,

String deptName, String hostelName, String hostelLocation, int numberOfRooms) {

setStudentName(studentName);

setElectiveSubject(electiveSubject);

setRegdNo(regdNo);

setAvgMarks(avgMarks);

setDeptName(deptName);

setDeptHead(deptHead);

super.setHostelName(hostelName);

super.setHostelLocation(hostelLocation);

super.setNumberOfRooms(numberOfRooms);

}

public void getData() {

System.out.println("Student Name: " + getStudentName());

System.out.println("Elective Subject: " + getElectiveSubject());

System.out.println("Registration Number: " + getregdNo());

System.out.println("Average Marks: " + getAvgMarks());

System.out.println("Hostel Name: " + super.getHostelName());

System.out.println("Hostel Location: " + super.getHostelLocation());

System.out.println("Number of Rooms: " + super.getNumberOfRooms());

}

@Override

public void Department() {

System.out.println("Department Name: " + getDeptName());

System.out.println("Department Head: " + getDeptHead());

}

}

public class q6 {

static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

System.out.println("Enter Number of Students: ");

int n = sc.nextInt();

student ar[] = new student[n];

int count = 0;

while (true) {

System.out.println("1. Admit New Student");

System.out.println("2. Migrate a Student");

System.out.println("3. Details of a Student");

System.out.println("4. Exit");

int ch = sc.nextInt();

switch (ch) {

case 1:

ar[count++] = new student();

System.out.println("Enter Student Name: ");

String studentName = sc.next();

System.out.println("Enter Elective Subject: ");

String electiveSubject = sc.next();

System.out.println("Enter Registration Number: ");

int regdNo = sc.nextInt();

System.out.println("Enter Average Marks: ");

double avgMarks = sc.nextDouble();

System.out.println("Enter Department Name: ");

String deptName = sc.next();

System.out.println("Enter Department Head: ");

String deptHead = sc.next();

System.out.println("Enter Hostel Name: ");

String hostelName = sc.next();

System.out.println("Enter Hostel Location: ");

String hostelLocation = sc.next();

System.out.println("Enter Number of Rooms: ");

int numberOfRooms = sc.nextInt();

ar[count - 1].setData(studentName, electiveSubject, regdNo, avgMarks, deptHead, deptName,

hostelName, hostelLocation, numberOfRooms);

System.out.println("Student Added");

break;

case 2:

System.out.println("Enter Registration Number of the Student: ");

int oldRegdNo = sc.nextInt();

student migrant = null;

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

if (ar[i].regdNo == oldRegdNo) {

migrant = ar[i];

ar[i] = null;

break;

}

}

if (migrant != null) {

while (true) {

System.out

.println("Do you want to Change any other Details of this Student?\n1. Yes\n2. No");

int x = sc.nextInt();

if (x == 1) {

System.out.println(

"a. Elective Subject\nb. Registration Number\nc. Average Marks\nd. Department\n");

char choice = sc.next().charAt(0);

switch (choice) {

case 'a':

System.out.println("Enter New Elective Subject: ");

migrant.setElectiveSubject(sc.next());

break;

case 'b':

System.out.println("Enter Registration Number: ");

migrant.setRegdNo(sc.nextInt());

break;

case 'c':

System.out.println("Enter New Average Marks: ");

migrant.setAvgMarks(sc.nextDouble());

break;

case 'd':

System.out.println("Enter New Department Name: ");

migrant.setDeptName(sc.next());

System.out.println("Enter New Department Head: ");

migrant.setDeptHead(sc.next());

break;

}

} else {

break;

}

}

int newIndex = -1;

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

if (ar[i] == null) {

newIndex = i;

break;

}

}

if (newIndex != -1) {

ar[newIndex] = migrant;

System.out.println("Student Migrated Successfully");

} else {

System.out.println("No Free Slots are Available Right Now");

}

} else {

System.out.println("Student Not Found");

}

break;

case 3:

System.out.print("Enter Registration Number: ");

int r = sc.nextInt();

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

if (ar[i].regdNo == r) {

ar[i].getData();

ar[i].Department();

break;

}

}

break;

case 4:

System.out.println("Exiting!");

System.exit(0);

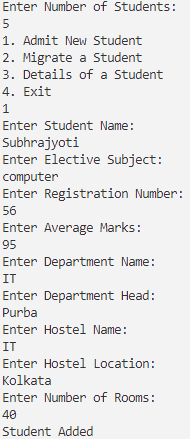
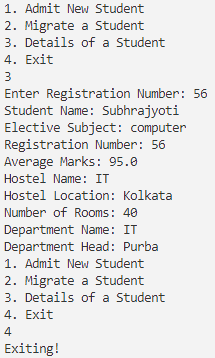
break;

}

}

}

}

**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.**

import java.util.Scanner;

interface Player {

void play();

}

class Child implements Player {

@Override

public void play() {

System.out.println("For a child, playing means having fun and enjoying activities.");

}

}

class Musician implements Player {

@Override

public void play() {

System.out.println("For a musician, playing means performing music with instruments or vocals.");

}

}

class Actor implements Player {

@Override

public void play() {

System.out.println("For an actor, playing means performing roles in theater, film, or television.");

}

}

public class q7\_player\_child\_musician\_actor {

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("Choose a player to learn about their definition of 'play':");

System.out.println("1. Child");

System.out.println("2. Musician");

System.out.println("3. Actor");

System.out.print("Enter your choice: ");

int choice = sc.nextInt();

sc.nextLine();

Player player;

switch (choice) {

case 1:

player = new Child();

break;

case 2:

player = new Musician();

break;

case 3:

player = new Actor();

break;

default:

System.out.println("Invalid choice.");

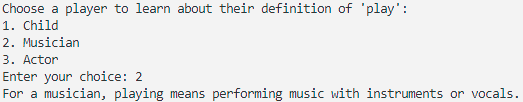
return;

}

player.play();

}

}



**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()**

import java.util.Scanner;

abstract class Accounts {

protected double balance;

protected int accountNumber;

protected String accountHolderName;

protected String address;

public Accounts(int accountNumber, String accountHolderName, String address) {

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.address = address;

this.balance = 0; // Initial balance is zero

}

public abstract void withdrawal(double amount);

public abstract void deposit(double amount);

public void display() {

System.out.println("Account Number: " + accountNumber);

System.out.println("Account Holder's Name: " + accountHolderName);

System.out.println("Address: " + address);

System.out.println("Current Balance: " + balance);

}

}

class SavingsAccount extends Accounts {

private double rateOfInterest;

public SavingsAccount(int accountNumber, String accountHolderName, String address, double rateOfInterest) {

super(accountNumber, accountHolderName, address);

this.rateOfInterest = rateOfInterest;

}

public void calculateAmount() {

double interest = balance \* (rateOfInterest / 100);

balance += interest;

}

@Override

public void withdrawal(double amount) {

if (balance >= amount) {

balance -= amount;

System.out.println("Withdrawal successful. Remaining balance: " + balance);

} else {

System.out.println("Insufficient funds.");

}

}

@Override

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit successful. Current balance: " + balance);

}

}

public class q8\_account\_savingsAccount {

public static void main(String[] args) {

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in);

System.out.println("Enter account number:");

int accountNumber = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.println("Enter account holder's name:");

String accountHolderName = scanner.nextLine();

System.out.println("Enter address:");

String address = scanner.nextLine();

System.out.println("Enter rate of interest:");

double rateOfInterest = scanner.nextDouble();

SavingsAccount savingsAccount = new SavingsAccount(accountNumber, accountHolderName, address, rateOfInterest);

System.out.println("Choose an option:");

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Display account details");

System.out.println("4. Exit");

while (true) {

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

double amount;

switch (choice) {

case 1:

System.out.println("Enter amount to deposit:");

amount = scanner.nextDouble();

savingsAccount.deposit(amount);

break;

case 2:

System.out.println("Enter amount to withdraw:");

amount = scanner.nextDouble();

savingsAccount.withdrawal(amount);

break;

case 3:

savingsAccount.display();

break;

case 4:

System.out.println("Exiting...");

System.exit(0);

default:

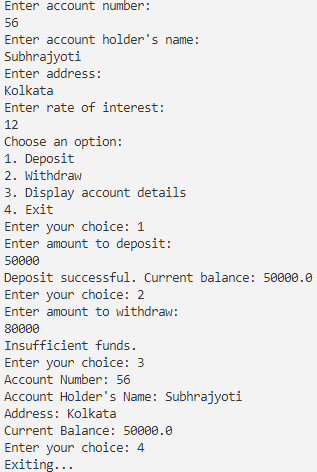
System.out.println("Invalid choice.");

}

}

}

}



**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.**

import java.util.Scanner;

abstract class MotorVehicle {

protected String modelName;

protected int modelNumber;

protected double modelPrice;

public MotorVehicle(String modelName, int modelNumber, double modelPrice) {

this.modelName = modelName;

this.modelNumber = modelNumber;

this.modelPrice = modelPrice;

}

abstract void display();

}

class Car extends MotorVehicle {

private double discountRate;

public Car(String modelName, int modelNumber, double modelPrice, double discountRate) {

super(modelName, modelNumber, modelPrice);

this.discountRate = discountRate;

}

@Override

void display() {

System.out.println("Car Name: " + modelName);

System.out.println("Model Number: " + modelNumber);

System.out.println("Price: $" + modelPrice);

System.out.println("Discount Rate: " + discountRate + "%");

}

public double discount() {

return modelPrice \* (discountRate / 100);

}

}

public class q9\_motorvehicle\_carthat {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Car details:");

System.out.print("Model Name: ");

String modelName = sc.nextLine();

System.out.print("Model Number: ");

int modelNumber = sc.nextInt();

System.out.print("Model Price: $");

double modelPrice = sc.nextDouble();

System.out.print("Discount Rate (%): ");

double discountRate = sc.nextDouble();

Car car = new Car(modelName, modelNumber, modelPrice, discountRate);

System.out.println("\nCar Details:");

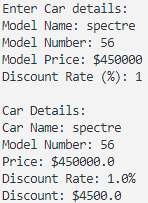
car.display();

System.out.println("Discount: $" + car.discount());

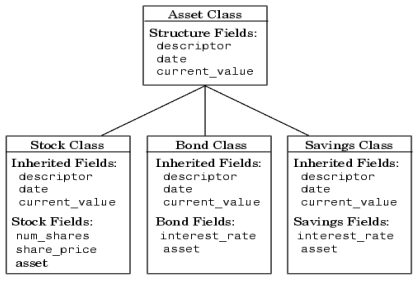
sc.close();

}

}



**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.**

****

import java.util.Scanner;

abstract class Asset {

protected String descriptor;

protected String date;

protected double 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) {

this.descriptor = descriptor;

this.date = date;

this.currentValue = currentValue;

this.numShares = numShares;

this.sharePrice = sharePrice;

}

public void displayDetails() {

System.out.println("Stock Details:");

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 double interestRate;

public Bond(String descriptor, String date, double currentValue, double interestRate) {

this.descriptor = descriptor;

this.date = date;

this.currentValue = currentValue;

this.interestRate = interestRate;

}

public void displayDetails() {

System.out.println("Bond Details:");

System.out.println("Descriptor: " + descriptor);

System.out.println("Date: " + date);

System.out.println("Current Value: " + currentValue);

System.out.println("Interest Rate: " + interestRate);

}

}

class Savings extends Asset {

private double interestRate;

public Savings(String descriptor, String date, double currentValue, double interestRate) {

this.descriptor = descriptor;

this.date = date;

this.currentValue = currentValue;

this.interestRate = interestRate;

}

public void displayDetails() {

System.out.println("Savings Details:");

System.out.println("Descriptor: " + descriptor);

System.out.println("Date: " + date);

System.out.println("Current Value: " + currentValue);

System.out.println("Interest Rate: " + interestRate);

}

}

public class q10\_asset\_stock\_bond\_saving {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter stock details:");

System.out.print("Descriptor: ");

String stockDescriptor = sc.nextLine();

System.out.print("Date: ");

String stockDate = sc.nextLine();

System.out.print("Current Value: ");

double stockValue = sc.nextDouble();

System.out.print("Number of Shares: ");

int numShares = sc.nextInt();

System.out.print("Share Price: ");

double sharePrice = sc.nextDouble();

Stock stock = new Stock(stockDescriptor, stockDate, stockValue, numShares, sharePrice);

System.out.println("\nEnter bond details:");

System.out.print("Descriptor: ");

sc.nextLine(); // Consume newline character

String bondDescriptor = sc.nextLine();

System.out.print("Date: ");

String bondDate = sc.nextLine();

System.out.print("Current Value: ");

double bondValue = sc.nextDouble();

System.out.print("Interest Rate: ");

double interestRate = sc.nextDouble();

Bond bond = new Bond(bondDescriptor, bondDate, bondValue, interestRate);

System.out.println("\nEnter savings details:");

System.out.print("Descriptor: ");

sc.nextLine(); // Consume newline character

String savingsDescriptor = sc.nextLine();

System.out.print("Date: ");

String savingsDate = sc.nextLine();

System.out.print("Current Value: ");

double savingsValue = sc.nextDouble();

System.out.print("Interest Rate: ");

double savingsInterestRate = sc.nextDouble();

Savings savings = new Savings(savingsDescriptor, savingsDate, savingsValue, savingsInterestRate);

System.out.println("\nDisplaying Details:");

stock.displayDetails();

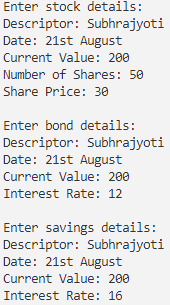
bond.displayDetails();

savings.displayDetails();

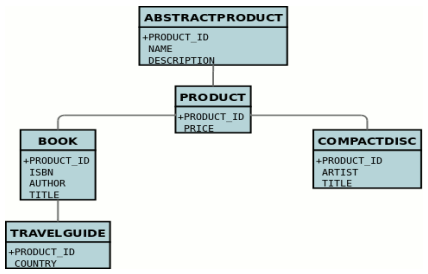
sc.close();

}

}

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



import java.util.Scanner;

abstract class AbstractProduct {

protected String productId;

protected String name;

protected String description;

public AbstractProduct(String productId, String name, String description) {

this.productId = productId;

this.name = name;

this.description = description;

}

public abstract void display();

}

class Product extends AbstractProduct {

public Product(String productId, String name, String description) {

super(productId, name, description);

}

@Override

public void display() {

System.out.println("Product ID: " + productId);

System.out.println("Name: " + name);

System.out.println("Description: " + description);

}

}

class Book extends Product {

private String isbn;

private String author;

public Book(String productId, String name, String description, String isbn, String author) {

super(productId, name, description);

this.isbn = isbn;

this.author = author;

}

@Override

public void display() {

super.display();

System.out.println("ISBN: " + isbn);

System.out.println("Author: " + author);

}

}

class CompactDisc extends Product {

private String artist;

private String title;

public CompactDisc(String productId, String name, String description, String artist, String title) {

super(productId, name, description);

this.artist = artist;

this.title = title;

}

@Override

public void display() {

super.display();

System.out.println("Artist: " + artist);

System.out.println("Title: " + title);

}

}

class TravelGuide extends Book {

private String country;

public TravelGuide(String productId, String name, String description, String isbn, String author, String country) {

super(productId, name, description, isbn, author);

this.country = country;

}

@Override

public void display() {

super.display();

System.out.println("Country: " + country);

}

}

public class q11\_abstractproduct\_product\_book\_compactdisc\_travelguide {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Travel Guide details:");

System.out.print("Product ID: ");

String productId = sc.nextLine();

System.out.print("Name: ");

String name = sc.nextLine();

System.out.print("Description: ");

String description = sc.nextLine();

System.out.print("ISBN: ");

String isbn = sc.nextLine();

System.out.print("Author: ");

String author = sc.nextLine();

System.out.print("Country: ");

String country = sc.nextLine();

TravelGuide travelGuide = new TravelGuide(productId, name, description, isbn, author, country);

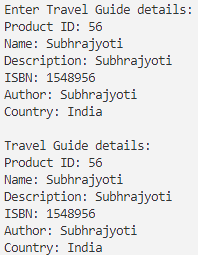
System.out.println("\nTravel Guide details:");

travelGuide.display();

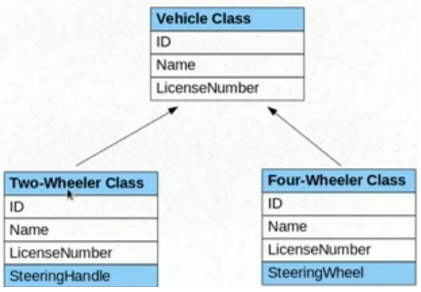
sc.close();

}

}



**Implement the below Diagram**



import java.util.Scanner;

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;

}

public int getID() {

return ID;

}

public String getName() {

return name;

}

public String getLicenseNumber() {

return 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;

}

public String getSteeringHandle() {

return steeringHandle;

}

}

class FourWheeler extends Vehicle {

private String steeringWheel;

public FourWheeler(int ID, String name, String licenseNumber, String steeringWheel) {

super(ID, name, licenseNumber);

this.steeringWheel = steeringWheel;

}

public String getSteeringWheel() {

return steeringWheel;

}

}

public class q12\_vehicle\_2\_4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Two-Wheeler details:");

System.out.print("ID: ");

int twoWheelerID = sc.nextInt();

sc.nextLine(); // Consume newline

System.out.print("Name: ");

String twoWheelerName = sc.nextLine();

System.out.print("License Number: ");

String twoWheelerLicenseNumber = sc.nextLine();

System.out.print("Steering Handle: ");

String twoWheelerSteeringHandle = sc.nextLine();

TwoWheeler twoWheeler = new TwoWheeler(twoWheelerID, twoWheelerName, twoWheelerLicenseNumber, twoWheelerSteeringHandle);

System.out.println("\nEnter Four-Wheeler details:");

System.out.print("ID: ");

int fourWheelerID = sc.nextInt();

sc.nextLine();

System.out.print("Name: ");

String fourWheelerName = sc.nextLine();

System.out.print("License Number: ");

String fourWheelerLicenseNumber = sc.nextLine();

System.out.print("Steering Wheel: ");

String fourWheelerSteeringWheel = sc.nextLine();

FourWheeler fourWheeler = new FourWheeler(fourWheelerID, fourWheelerName, fourWheelerLicenseNumber, fourWheelerSteeringWheel);

System.out.println("\nTwo-Wheeler Details:");

System.out.println("ID: " + twoWheeler.getID());

System.out.println("Name: " + twoWheeler.getName());

System.out.println("License Number: " + twoWheeler.getLicenseNumber());

System.out.println("Steering Handle: " + twoWheeler.getSteeringHandle());

System.out.println("\nFour-Wheeler Details:");

System.out.println("ID: " + fourWheeler.getID());

System.out.println("Name: " + fourWheeler.getName());

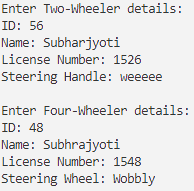
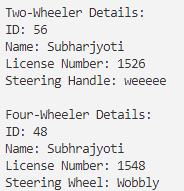
System.out.println("License Number: " + fourWheeler.getLicenseNumber());

System.out.println("Steering Wheel: " + fourWheeler.getSteeringWheel());

sc.close();

}

}

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

import java.util.Scanner;

interface Bank {

void getBankDetails();

}

interface Customer {

void getCustomerDetails();

}

class Account implements Bank, Customer {

private String bankName;

private String customerName;

private String accountNumber;

static Scanner sc = new Scanner(System.in);

public void getBankDetails() {

System.out.print("Enter Bank Name: ");

bankName = sc.nextLine();

}

public void getCustomerDetails() {

System.out.print("Enter Customer Name: ");

customerName = sc.nextLine();

System.out.print("Enter Account Number: ");

accountNumber = sc.nextLine();

}

public void displayDetails() {

System.out.println("\nBank Name: " + bankName);

System.out.println("Customer Name: " + customerName);

System.out.println("Account Number: " + accountNumber);

}

}

public class q13\_multiple\_inheritance\_bank {

public static void main(String[] args) {

Account account = new Account();

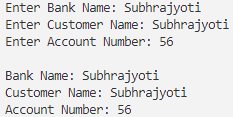
account.getBankDetails();

account.getCustomerDetails();

account.displayDetails();

}

}



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

import java.util.Scanner;

interface Gross {

double calculateGross();

}

interface Employee {

void getEmployeeDetails();

}

class Salary implements Gross, Employee {

private String name;

private double basicSalary;

private double allowances;

static Scanner sc = new Scanner(System.in);

public void getEmployeeDetails() {

System.out.print("Enter employee name: ");

name = sc.nextLine();

System.out.print("Enter basic salary: ");

basicSalary = sc.nextDouble();

System.out.print("Enter allowances: ");

allowances = sc.nextDouble();

}

public double calculateGross() {

return basicSalary + allowances;

}

public void displayDetails() {

System.out.println("\nEmployee Name: " + name);

System.out.println("Basic Salary: " + basicSalary);

System.out.println("Allowances: " + allowances);

System.out.println("Gross Salary: " + calculateGross());

}

}

public class q14\_multiple\_inheritance\_gross {

public static void main(String[] args) {

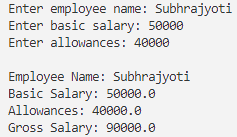
Salary employee = new Salary();

employee.getEmployeeDetails();

employee.displayDetails();

}

}



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

interface Mango {

void displayType();

}

class Winter implements Mango {

public void displayType() {

System.out.println("This is a Winter Mango.");

}

}

class Summer implements Mango {

public void displayType() {

System.out.println("This is a Summer Mango.");

}

}

public class q15\_mango\_summer\_winter {

public static void main(String[] args) {

Winter winterMango = new Winter();

Summer summerMango = new Summer();

winterMango.displayType();

summerMango.displayType();

}

}



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

import java.util.Scanner;

interface Exam {

double percent\_cal(double marks1, double marks2);

}

class Student {

String name;

int rollNo;

double marks1;

double marks2;

public Student(String name, int rollNo, double marks1, double marks2) {

this.name = name;

this.rollNo = rollNo;

this.marks1 = marks1;

this.marks2 = marks2;

}

public void show() {

System.out.println("Name: " + name);

System.out.println("Roll No: " + rollNo);

System.out.println("Marks 1: " + marks1);

System.out.println("Marks 2: " + marks2);

}

}

class Result extends Student implements Exam {

public Result(String name, int rollNo, double marks1, double marks2) {

super(name, rollNo, marks1, marks2);

}

@Override

public double percent\_cal(double marks1, double marks2) {

return ((marks1 + marks2) / 200) \* 100;

}

public void display() {

super.show();

double percentage = percent\_cal(marks1, marks2);

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

}

}

public class q16\_stu\_exm\_res {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter student name:");

String name = sc.nextLine();

System.out.println("Enter student roll number:");

int rollNo = sc.nextInt();

System.out.println("Enter marks for subject 1:");

double marks1 = sc.nextDouble();

System.out.println("Enter marks for subject 2:");

double marks2 = sc.nextDouble();

Result result = new Result(name, rollNo, marks1, marks2);

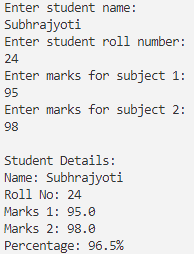
System.out.println("\nStudent Details:");

result.display();

sc.close();

}

}



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

interface Animal {

void eat();

}

interface Mammal extends Animal {

void walk();

}

interface Reptile extends Animal {

void crawl();

}

class Dog implements Mammal {

public void eat() {

System.out.println("Dog is eating.");

}

public void walk() {

System.out.println("Dog is walking.");

}

}

class Snake implements Reptile {

public void eat() {

System.out.println("Snake is eating.");

}

public void crawl() {

System.out.println("Snake is crawling.");

}

}

public class q17\_hierarchial\_inheritance {

public static void main(String[] args) {

Dog dog = new Dog();

Snake snake = new Snake();

dog.eat();

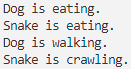
snake.eat();

dog.walk();

snake.crawl();

}

}



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

import java.util.Scanner;

interface Payable {

double calculateBasicSalary();}

interface Taxable {

double calculateTax();}

class Employee implements Payable, Taxable {

private double basicSalary;

private double taxRate;

public Employee(double basicSalary, double taxRate) {

this.basicSalary = basicSalary;

this.taxRate = taxRate;}

public double calculateBasicSalary() {

return basicSalary;}

public double calculateTax() {

return basicSalary \* (taxRate / 100);}}

public class q18\_multiple\_inheritance\_payroll\_emp

{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the employee's basic salary: ");

double basicSalary = scanner.nextDouble();

System.out.print("Enter the tax rate (in percentage): ");

double taxRate = scanner.nextDouble();

Employee employee = new Employee(basicSalary, taxRate);

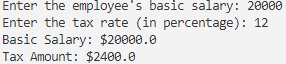
System.out.println("Basic Salary: $" + employee.calculateBasicSalary());

System.out.println("Tax Amount: $" + employee.calculateTax());

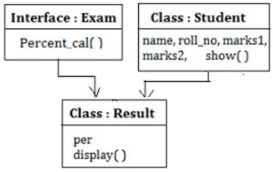
scanner.close();

}

}



**Implement the following diagram.**



import java.util.Scanner;

interface Exam {

void conductExam();

}

class Student {

private String name;

private int rollNumber;

public Student(String name, int rollNumber) {

this.name = name;

this.rollNumber = rollNumber;

}

public void displayDetails() {

System.out.println("Student Name: " + name);

System.out.println("Roll Number: " + rollNumber);

}

}

class Result implements Exam {

private int marks;

public void conductExam() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter marks obtained: ");

marks = sc.nextInt();

sc.close();

}

public void displayResult() {

System.out.println("Marks Obtained: " + marks);

}

}

public class q19\_student\_exam\_result {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter student name: ");

String name = sc.nextLine();

System.out.print("Enter roll number: ");

int rollNumber = sc.nextInt();

Student student = new Student(name, rollNumber);

System.out.println("\nConducting exam for student...");

Result result = new Result();

result.conductExam();

System.out.println("\nStudent Details:");

student.displayDetails();

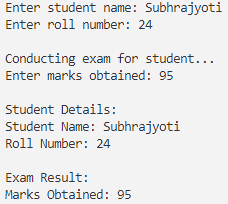
System.out.println("\nExam Result:");

result.displayResult();

sc.close();

}

}



WEEK – 7

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

public class Q\_1 {

public static void main(String[] args) {

try {

int result = divide(10, 0);

System.out.println("Result: " + result);

} catch (ArithmeticException e) {

System.out.println("Exception occurred: Division by zero");

} finally {

System.out.println("This block always executes.");

}

throwExample();

}

public static int divide(int a, int b) throws ArithmeticException {

if (b == 0) {

throw new ArithmeticException("Division by zero");

}

return a / b;

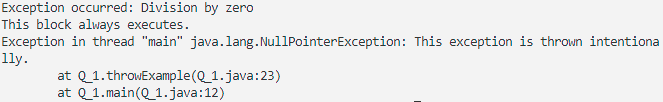
}

public static void throwExample() {

throw new NullPointerException("This exception is thrown intentionally.");

}

}



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

public class Q\_2 {

public static void main(String[] args) {

try {

int[] arr = {1, 2, 3};

int index = -1;

int element = arr[index];

System.out.println("Element at index " + index + ": " + element);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException occurred: " + e.getMessage());

}

try {

int result = 5 / 0;;

System.out.println("Result: " + result);

} catch (ArithmeticException e) {

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

}

}

}



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

import java.util.Scanner;

public class Q\_3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String inputStr = scanner.nextLine();

scanner.close();

try {

checkString(inputStr);

System.out.println("Match found!");

} catch (NoMatchFoundException e) {

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

}

}

static void checkString(String inputStr) throws NoMatchFoundException {

if (!inputStr.equals("University")) {

throw new NoMatchFoundException("Input string does not match 'University'");

}

}

}

class NoMatchFoundException extends Exception {

public NoMatchFoundException(String message) {

super(message);

}

}



**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.**

public class Q\_4 {

private int totalAlphabeticCharacters;

public Q\_4() {

this.totalAlphabeticCharacters = 0;

}

public void addCharacter(char c) throws IllegalArgumentException {

if (!Character.isLetter(c)) {

throw new IllegalArgumentException("Non-alphabetic character passed: " + c);

}

totalAlphabeticCharacters++;

}

public int getTotalAlphabeticCharacters() {

return totalAlphabeticCharacters;

}

public static void main(String[] args) {

Q\_4 counter = new Q\_4();

try {

counter.addCharacter('a');

counter.addCharacter('b');

counter.addCharacter('1');

} catch (IllegalArgumentException e) {

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

}

System.out.println("Total alphabetic characters: " + counter.getTotalAlphabeticCharacters());

}

}



**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.**

public class Q\_5 {

private static final int MAX\_SIZE = 21;

private static long[] factorialArray = new long[MAX\_SIZE];

static {

factorialArray[0] = 1;

for (int i = 1; i < MAX\_SIZE; i++) {

factorialArray[i] = factorialArray[i - 1] \* i;

}

}

public static long computeFactorial(int x) {

if (x < 0) {

throw new IllegalArgumentException("Value of x must be positive");

}

if (x >= MAX\_SIZE) {

throw new IllegalArgumentException("Result will overflow");

}

return factorialArray[x];

}

public static void main(String[] args) {

try {

int x = 21;

long result = computeFactorial(x);

System.out.println("Factorial of " + x + " is: " + result);

} catch (IllegalArgumentException e) {

System.out.println("Exception occurred: " + e.getMessage());

}

}

}



**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.**

class CharacterCounter {

private int totalCount;

CharacterCounter() {

this.totalCount = 0;

}

public void countCharacter(char c) throws Exception {

if (!Character.isAlphabetic(c)) {

throw new Exception("Non-alphabetic character encountered: " + c);

}

totalCount++;

}

public int getTotalCount() {

return totalCount;

}

}

public class Q\_6 {

public static void main(String[] args) {

CharacterCounter counter = new CharacterCounter();

String input = "Subhrajyoti Sakar 007";

for (int i = 0; i < input.length(); i++) {

char c = input.charAt(i);

try {

counter.countCharacter(c);

} catch (Exception e) {

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

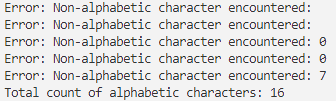
}

}

System.out.println("Total count of alphabetic characters: " + counter.getTotalCount());

}

}



**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.**

import java.util.HashMap;

import java.util.Map;

public class Q\_7 {

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Please provide a country name as a command line argument.");

return;

}

String countryName = args[0];

Map<String, String> capitals = createCapitalMap();

try {

String capital = capitals.get(countryName);

if (capital == null) {

throw new NoMatchFoundException("No capital found for country: " + countryName);

}

System.out.println("The capital of " + countryName + " is " + capital);

} catch (NoMatchFoundException e) {

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

}

}

private static Map<String, String> createCapitalMap() {

Map<String, String> capitals = new HashMap<>();

capitals.put("France", "Paris");

capitals.put("Germany", "Berlin");

capitals.put("Italy", "Rome");

capitals.put("Spain", "Madrid");

return capitals;

}

public static class NoMatchFoundException extends Exception {

public NoMatchFoundException(String message) {

super(message);

}

}

}



**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.**

public class Q\_8 {

public static void main(String[] args) {

try {

if (args.length == 0) {

throw new ArrayIndexOutOfBoundsException("No argument provided. Please enter an integer.");

}

int number = Integer.parseInt(args[0]);

if (number == 0) {

throw new IllegalArgumentException("Factorial of 0 is not allowed.");

}

int factorial = 1;

for (int i = 1; i <= number; i++) {

factorial \*= i;

}

System.out.println("Factorial of " + number + " is: " + factorial);

} catch (ArrayIndexOutOfBoundsException e) {

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

} catch (NumberFormatException e) {

System.out.println("Error: Please enter a valid integer.");

} catch (IllegalArgumentException e) {

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

}

}

}



**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.**

class CheckArgument extends Exception {

public CheckArgument(String message) {

super(message);

}

}

public class Q\_9 {

public static void main(String[] args) {

try {

if (args.length < 5) {

throw new CheckArgument("Insufficient arguments. Please provide at least 5 numbers.");

}

int sum = 0;

for (int i = 0; i < 5; i++) {

sum += Integer.parseInt(args[i]);

}

System.out.println("The sum of the first five numbers is: " + sum);

} catch (CheckArgument e) {

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

} catch (NumberFormatException e) {

System.out.println("Error: Please ensure all arguments are integers.");

}

}

}



**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.**

public class Q\_10 {

public static void main(String[] args) {

try {

if (args.length != 7) {

throw new IllegalArgumentException("Please provide the student's name and marks for six subjects.");

}

String studentName = args[0];

int totalMarks = 0;

int marks;

for (int i = 1; i <= 6; i++) {

marks = Integer.parseInt(args[i]);

if (marks < 0 || marks > 50) {

throw new RangeException("Marks for subject " + i + " are out of range.");

}

totalMarks += marks;

}

double percentage = (totalMarks / 300.0) \* 100;

System.out.println("Mark Sheet for " + studentName);

System.out.println("Total Marks: " + totalMarks);

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

} catch (NumberFormatException e) {

System.out.println("Please enter valid integer marks.");

} catch (RangeException e) {

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

} catch (Exception e) {

System.out.println("An unexpected error occurred: " + e.getMessage());

}

}

static class RangeException extends Exception {

public RangeException(String message) {

super(message);

}

}

}



**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).**

public class Q\_11 {

public static void main(String[] args) {

try {

int X = 10;

int Y = 20;

int P = 30;

int Q = 0;

int Z = 40;

int I = 50;

int result = calculateEquation(X, Y, P, Q, Z, I);

System.out.println("The result of the equation is: " + result);

} catch (ArithmeticExceptionHandler e) {

System.out.println("Arithmetic Exception Occurred: " + e.getMessage());

}

}

public static int calculateEquation(int X, int Y, int P, int Q, int Z, int I) throws ArithmeticExceptionHandler {

if (Q == 0) {

throw new ArithmeticExceptionHandler("Cannot divide by zero");

}

if (P % Q != 0) {

throw new ArithmeticExceptionHandler("Invalid operation: Non-integer division");

}

return X + Y \* (P / Q) \* Z - I;

}

static class ArithmeticExceptionHandler extends Exception {

public ArithmeticExceptionHandler(String message) {

super(message);

}

}

}



**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.**

public class Q\_12 {

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Please provide a temperature value in Celsius as a command line argument.");

return;

}

double celsius;

try {

celsius = Double.parseDouble(args[0]);

} catch (NumberFormatException e) {

System.out.println("Invalid temperature format. Please enter a number.");

return;

}

try {

checkTemperature(celsius);

} catch (TooHotException e) {

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

} catch (TooColdException e) {

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

}

}

public static void checkTemperature(double celsius) throws TooHotException, TooColdException {

if (celsius > 35) {

throw new TooHotException("Temperature is too hot!");

} else if (celsius < 5) {

throw new TooColdException("Temperature is too cold!");

} else {

double fahrenheit = (celsius \* 9 / 5) + 32;

System.out.println("Normal temperature");

System.out.printf("Temperature in Fahrenheit: %.2f\n", fahrenheit);

}

}

public static class TooHotException extends Exception {

public TooHotException(String message) {

super(message);

}

}

public static class TooColdException extends Exception {

public TooColdException(String message) {

super(message);

}

}

}



**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.**

import java.util.Scanner;

public class Q\_13 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter candidate name: ");

String name = scanner.nextLine();

System.out.print("Enter candidate age: ");

int age = scanner.nextInt();

try {

validateAge(age);

System.out.println("Eligible candidate: " + name);

} catch (TooOlderException e) {

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

} catch (TooYoungerException e) {

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

}

scanner.close();

}

public static void validateAge(int age) throws TooOlderException, TooYoungerException {

if (age > 45) {

throw new TooOlderException("Candidate is too old for the position.");

} else if (age < 20) {

throw new TooYoungerException("Candidate is too young for the position.");

}

}

public static class TooOlderException extends Exception {

public TooOlderException(String message) {

super(message);

}

}

public static class TooYoungerException extends Exception {

public TooYoungerException(String message) {

super(message);

}

}

}



**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.**

import java.util.Scanner;

public class Q\_14 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a binary number: ");

String binaryString = scanner.nextLine();

try {

int decimal = convertBinaryToDecimal(binaryString);

System.out.println("Decimal equivalent: " + decimal);

} catch (WrongNumberFormatException e) {

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

}

scanner.close();

}

public static int convertBinaryToDecimal(String binaryString) throws WrongNumberFormatException {

int decimal = 0;

int base = 1;

for (int i = binaryString.length() - 1; i >= 0; i--) {

char digit = binaryString.charAt(i);

if (digit != '0' && digit != '1') {

throw new WrongNumberFormatException("Invalid binary digit. Only '0' and '1' allowed.");

}

int digitValue = Character.getNumericValue(digit);

decimal += digitValue \* base;

base \*= 2;

}

return decimal;

}

public static class WrongNumberFormatException extends Exception {

public WrongNumberFormatException(String message) {

super(message);

}

}

}



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

public class Q\_15 {

public static void main(String[] args) {

try {

int[] numbers = {1, 2, 3};

System.out.println("Before exception is generated.");

try {

System.out.println(numbers[5]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array index is out of bounds: " + e);

}

System.out.println(5 / 0);

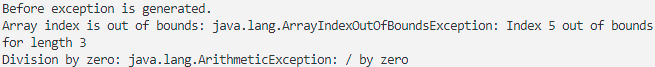
} catch (ArithmeticException e) {

System.out.println("Division by zero: " + e);

}

}

}



**Write a Java Program to Create Account with 500 Rs Minimum Balance, Deposit Amount, Withdraw Amount and Also Throws LessBalanceException.**

* **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.**

class LessBalanceException extends Exception {

public LessBalanceException(String message) {

super(message);

}

}

class Account {

private static final int MIN\_BALANCE = 500;

private int balance;

public Account(int initialDeposit) throws LessBalanceException {

if (initialDeposit < MIN\_BALANCE) {

throw new LessBalanceException("Initial deposit must be at least Rs " + MIN\_BALANCE);

}

this.balance = initialDeposit;

}

public void deposit(int amount) {

balance += amount;

System.out.println("Deposited Rs " + amount + ". Current Balance: Rs " + balance);

}

public void withdraw(int amount) throws LessBalanceException {

if (amount > balance) {

throw new LessBalanceException("Withdrawal amount Rs " + amount + " is not valid. Current Balance: Rs " + balance);

}

balance -= amount;

System.out.println("Withdrawn Rs " + amount + ". Current Balance: Rs " + balance);

}

public int getBalance() {

return balance;

}

}

public class Q\_16 {

public static void main(String[] args) {

try {

Account account1 = new Account(1000);

Account account2 = new Account(1500);

account1.deposit(500);

account1.withdraw(200);

account2.deposit(400);

account2.withdraw(2000);

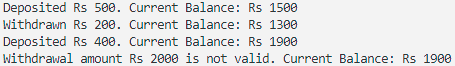
} catch (LessBalanceException e) {

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

}

}

}



**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”.**

class BookNotFoundException extends Exception {

public BookNotFoundException(String message) {

super(message);

}

}

class Library {

private String[] availableBooks;

public Library(String[] books) {

this.availableBooks = books;

}

public void findBook(String bookName) throws BookNotFoundException {

for (String book : availableBooks) {

if (book.equalsIgnoreCase(bookName)) {

System.out.println("Book found: " + book);

return;

}

}

throw new BookNotFoundException("BookNotFoundException: '" + bookName + "' does not exist in the library.");

}

}

public class Q\_17 {

public static void main(String[] args) {

String[] books = {"The Alchemist", "The Da Vinci Code", "Harry Potter", "The Lord of the Rings"};

Library library = new Library(books);

try {

library.findBook("The Da Vinci Code");

library.findBook("The Great Gatsby");

} catch (BookNotFoundException e) {

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

}

}

}



**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”.**

import java.util.Scanner;

class NotCorrectException extends Exception {

public NotCorrectException(String message) {

super(message);

}

}

class QuizManagementSystem {

static Scanner scanner = new Scanner(System.in);

private String[] questions = {

"What is the capital of France?",

"Who wrote 'Romeo and Juliet'?",

"What is the largest ocean on Earth?",

"What is the result of 7 \* 6?",

"What year did World War II end?"

};

private String[] answers = {

"Paris",

"William Shakespeare",

"Pacific",

"42",

"1945"

};

public void checkAnswer(int questionNumber, String answer) throws NotCorrectException {

if (answers[questionNumber].equalsIgnoreCase(answer)) {

System.out.println("Good! The answer is correct.");

} else {

throw new NotCorrectException("NotCorrectException: The answer is incorrect.");

}

}

public void startQuiz() {

for (int i = 0; i < questions.length; i++) {

System.out.println((i + 1) + ". " + questions[i]);

String userAnswer = getUserAnswer();

try {

checkAnswer(i, userAnswer);

} catch (NotCorrectException e) {

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

break;

}

}

}

private String getUserAnswer() {

System.out.print("Your answer: ");

String answer = scanner.nextLine();

return answer;

}

}

public class Q\_18 {

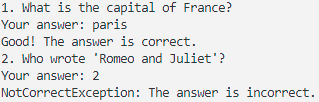
public static void main(String[] args) {

QuizManagementSystem quiz = new QuizManagementSystem();

quiz.startQuiz();

}

}



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

import java.util.Scanner;

public class Q\_19 {

private static final String VALID\_USERNAME = "SubhrajyotiSarkar007";

private static final String VALID\_PASSWORD = "SubhrajyotiSarkar";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter password: ");

String password = scanner.nextLine();

try {

authenticate(username, password);

System.out.println("Authentication successful!");

} catch (InvalidCredentialsException e) {

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

}

scanner.close();

}

private static void authenticate(String username, String password) throws InvalidCredentialsException {

if (username.length() < 6) {

throw new InvalidCredentialsException("Username must be at least 6 characters long.");

}

if (!username.equals(VALID\_USERNAME) || !password.equals(VALID\_PASSWORD)) {

throw new InvalidCredentialsException("Invalid username or password.");

}

}

}

class InvalidCredentialsException extends Exception {

public InvalidCredentialsException(String message) {

super(message);

}

}



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

import java.util.Scanner;

class AuthenticationFailureException extends Exception {

public AuthenticationFailureException(String message) {

super(message);

}

}

public class Q\_20 {

private static final String CORRECT\_PASSWORD = "SubhrajyotiSarkar";

public static void authenticate(String password) throws AuthenticationFailureException {

if (!password.equals(CORRECT\_PASSWORD)) {

throw new AuthenticationFailureException("Authentication Failure: Incorrect password");

}

System.out.println("Authentication Successful");

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter password: ");

String password = scanner.nextLine();

scanner.close();

try {

authenticate(password);

} catch (AuthenticationFailureException e) {

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

}

}

}



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

import java.util.Scanner;

class NegativeAgeException extends Exception {

public NegativeAgeException(String message) {

super(message);

}

}

public class Q\_21 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter name: ");

String name = scanner.nextLine();

System.out.print("Enter age: ");

int age = scanner.nextInt();

try {

validateAge(age);

System.out.println("Name: " + name);

System.out.println("Age: " + age);

} catch (NegativeAgeException e) {

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

}

scanner.close();

}

private static void validateAge(int age) throws NegativeAgeException {

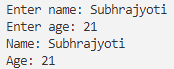
if (age < 0) {

throw new NegativeAgeException("Age cannot be negative.");

}

}

}



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

class NotPositiveException extends Exception {

public NotPositiveException(String message) {

super(message);

}

}

class Q\_22 {

public static void checkNumber(int number) throws NotPositiveException {

if (number <= 0) {

throw new NotPositiveException("Number is not positive.");

} else {

System.out.println(number + " is a positive number.");

}

}

public static void main(String[] args) {

try {

checkNumber(-5);

} catch (NotPositiveException e) {

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

}

}

}



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

import java.util.Scanner;

class StringMismatchException extends Exception {

public StringMismatchException(String message) {

super(message);

}

}

public class Q\_23 {

public static void compareStrings(String str1, String str2) throws StringMismatchException {

if (!str1.equalsIgnoreCase(str2)) {

throw new StringMismatchException("The strings do not match.");

} else {

System.out.println("The strings match.");

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the first string:");

String string1 = scanner.nextLine();

System.out.println("Enter the second string:");

String string2 = scanner.nextLine();

try {

compareStrings(string1, string2);

} catch (StringMismatchException e) {

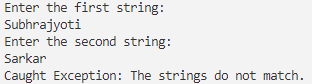
System.out.println("Caught Exception: " + e.getMessage());

}

scanner.close();

}

}



**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.**

class PushException extends Exception {

public PushException(String message) {

super(message);

}

}

class PopException extends Exception {

public PopException(String message) {

super(message);

}

}

class Stack {

private int maxSize;

private int[] stackArray;

private int top;

public Stack(int size) {

maxSize = size;

stackArray = new int[maxSize];

top = -1;

}

public void push(int value) throws PushException {

if (isFull()) {

throw new PushException("Stack is full. Cannot push " + value);

}

stackArray[++top] = value;

}

public int pop() throws PopException {

if (isEmpty()) {

throw new PopException("Stack is empty. Cannot pop.");

}

return stackArray[top--];

}

public boolean isEmpty() {

return (top == -1);

}

public boolean isFull() {

return (top == maxSize - 1);

}

}

class Q\_24 {

public static void main(String[] args) {

Stack stack = new Stack(3);

try {

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

System.out.println(stack.pop());

System.out.println(stack.pop());

System.out.println(stack.pop());

System.out.println(stack.pop());

} catch (PushException | PopException e) {

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

}

}

}



**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.**

import java.util.Scanner;

class ScoreException extends Exception {

public ScoreException(String message) {

super(message);

}

}

public class Q\_25 {

public static void main(String[] args) {

final int NUM\_STUDENTS = 5;

int[] studentIDs = { 101, 102, 103, 104, 105 };

int[] testScores = new int[NUM\_STUDENTS];

Scanner scanner = new Scanner(System.in);

for (int i = 0; i < NUM\_STUDENTS; i++) {

System.out.print("Enter test score for student with ID " + studentIDs[i] + ": ");

try {

int score = scanner.nextInt();

if (score < 0 || score > 100) {

throw new ScoreException("Invalid score. Score must be between 0 and 100.");

}

testScores[i] = score;

} catch (ScoreException e) {

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

testScores[i] = 0;

} catch (Exception e) {

System.out.println("Invalid input. Please enter a numeric value.");

scanner.next();

testScores[i] = 0;

}

}

System.out.println("\nStudent IDs and Scores:");

for (int i = 0; i < NUM\_STUDENTS; i++) {

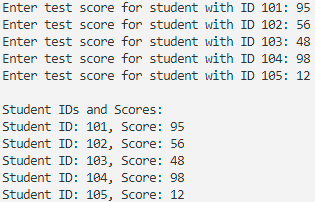
System.out.println("Student ID: " + studentIDs[i] + ", Score: " + testScores[i]);

}

scanner.close();

}

}



WEEK – 8

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

import java.util.Scanner;

public class q1\_factorial {

public static int calculateFactorial(int n) {

if (n == 0 || n == 1)

return 1;

return n \* calculateFactorial(n - 1);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int fact = calculateFactorial(number);

System.out.printf("Factorial of %d is: %d", number, fact);

scanner.close();

}

}



import java.io.BufferedReader;

import java.io.InputStreamReader;

public class q1\_2nd {

public static int calculateFactorial(int n) {

if (n == 0 || n == 1)

return 1;

return n \* calculateFactorial(n - 1);

}

public static void main(String[] args) throws Exception {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter a number: ");

String input = reader.readLine();

int number = Integer.parseInt(input);

int factorial = calculateFactorial(number);

System.out.println(factorial);

}

}



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

import java.util.Scanner;

class Palindrome {

String str = "";

Palindrome(String s) {

this.str = s;

}

public boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

}

public class q2\_palindrome {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String name = sc.nextLine();

Palindrome pd = new Palindrome(name);

if (pd.isPalindrome(name)) {

System.out.println("Palindrome");

} else {

System.out.println("Not Palindrome");

}

sc.close();

}

}



**Write a Java program to merge two strings.**

import java.util.Scanner;

public class q3\_merge\_strings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter two strings : ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

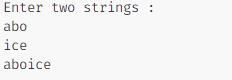
String merged = str1 + str2;

System.out.println(merged);

sc.close();

}

}



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

import java.util.Scanner;

public class q4\_reverse\_string {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.err.println("Enter a string : ");

String str = sc.nextLine();

String revStr = "";

for (int i = str.length() - 1; i >= 0; i--) {

revStr += str.charAt(i);

}

System.out.println(revStr);

sc.close();

}

}



**Write a Java Program to Concatenate Two Strings.**

import java.util.Scanner;

public class q5\_concatenate\_strings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter two strings : ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

StringBuilder sb = new StringBuilder();

sb.append(str1);

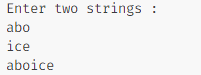
sb.append(str2);

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

sc.close();

}

}



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

import java.util.Scanner;

public class q6\_getChar {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

System.out.print("Enter Index: ");

int index = sc.nextInt();

if (checkChar(str, index)) {

System.out.println("The character at index " + index + " is '" + str.charAt(index) + "'");

} else {

System.out.println("Character does not exist at index " + index);

}

sc.close();

}

public static boolean checkChar(String str, int index) {

if (index < 0 || index >= str.length()) {

return false;

}

return true;

}

}



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

import java.util.Scanner;

public class q7\_length\_of\_string {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Length is " + str.length());

sc.close();

}

}



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

import java.util.\*;

public class q8\_subsets\_of\_string {

public static List<String> findSubsets(String str, int k) {

List<String> subsets = new ArrayList<>();

generateSubsets(str, 0, k, "", subsets);

return subsets;

}

private static void generateSubsets(String str, int index, int k, String current, List<String> subsets) {

if (current.length() == k) {

subsets.add(current);

return;

}

if (index == str.length())

return;

generateSubsets(str, index + 1, k, current + str.charAt(index), subsets);

generateSubsets(str, index + 1, k, current, subsets);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String input = sc.nextLine();

System.out.print("Enter Length of Substring: ");

int k = sc.nextInt();

List<String> subsets = findSubsets(input, k);

System.out.println("Subsets of length " + k + " in the string \"" + input + "\":");

for (String subset : subsets) {

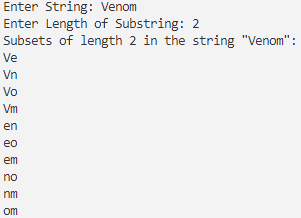
System.out.println(subset);

}

sc.close();

}

}



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

import java.util.Scanner;

public class q9\_remove\_white\_spaces {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

str = str.trim();

str = str.replaceAll("\\s", "");

System.out.println(str);

sc.close();

}

}



**Write a Java Program to Compare two Strings.**

import java.util.Scanner;

public class q10\_string\_compare {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter two strings : ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

boolean areEqual = str1.equalsIgnoreCase(str2);

System.out.println(areEqual);

sc.close();

}

public void transfer(q10\_string\_compare acc2, int i) {

throw new UnsupportedOperationException("Unimplemented method 'transfer'");

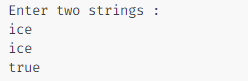
}

public String getBalance() {

throw new UnsupportedOperationException("Unimplemented method 'getBalance'");

}

}



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

import java.util.\*;

public class q11\_compare\_strings {

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

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

String str1 = sc.next();

System.out.print("Enter another String: ");

String str2 = sc.next();

// Method 1: Using equals() method

long startTime1 = System.nanoTime();

boolean isEqual1 = str1.equals(str2);

long endTime1 = System.nanoTime();

long executionTime1 = endTime1 - startTime1;

// Method 2: Using compareTo() method

long startTime2 = System.nanoTime();

int comparisonResult = str1.compareTo(str2);

boolean isEqual2 = comparisonResult == 0;

long endTime2 = System.nanoTime();

long executionTime2 = endTime2 - startTime2;

// Displaying results

System.out.println("Method 1 (equals()):");

System.out.println("Strings are equal: " + isEqual1);

System.out.println("Execution time: " + executionTime1 + " nanoseconds\n");

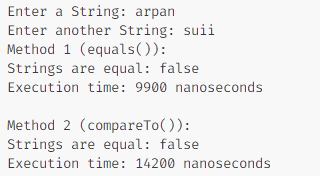
System.out.println("Method 2 (compareTo()):");

System.out.println("Strings are equal: " + isEqual2);

System.out.println("Execution time: " + executionTime2 + " nanoseconds");

}

}



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

import java.util.Scanner;

public class q12\_string\_equals {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Two Strings: ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

boolean areEqual = str1.equals(str2);

System.out.println("Using equals(): " + str1 + " == " + str2 + " is " + areEqual);

sc.close();

}

}



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

import java.util.\*;

class q13\_string\_equalsIgnoreCase {

static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

System.out.print("Enter String: ");

String s = sc.nextLine();

System.out.print("Enter Another String: ");

String t = sc.nextLine();

if (s.equalsIgnoreCase(t)) {

System.out.println(s + " and " + t + " are equal if we ignore the cases");

} else {

System.out.printf("%s and %s are not equal", s, t);

}

}

}



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

import java.util.Scanner;

public class q14\_compareTo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Two Strings: ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

int result1 = str1.compareTo(str2);

int result2 = str2.compareTo(str1);

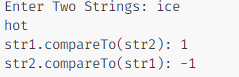
System.out.println("str1.compareTo(str2): " + result1);

System.out.println("str2.compareTo(str1): " + result2);

sc.close();

}

}



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

import java.util.Scanner;

public class q15\_compareToIgnoreCase {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter 2 Strings: ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

int result1 = str1.compareToIgnoreCase(str2);

int result2 = str2.compareToIgnoreCase(str1);

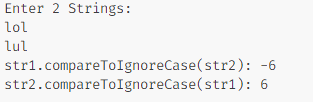
System.out.println("str1.compareToIgnoreCase(str2): " + result1);

System.out.println("str2.compareToIgnoreCase(str1): " + result2);

sc.close();

}

}



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

import java.util.Scanner;

public class q16\_replace\_character {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Enter an old and new char to change : ");

String old = sc.next();

String neww = sc.next();

String replacedString = str.replace(old, neww);

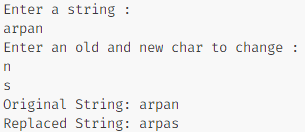
System.out.println("Original String: " + str);

System.out.println("Replaced String: " + replacedString);

sc.close();

}

}



**Write a Java Program to Search Last Occurrence of a Substring Inside a Substring.**

import java.util.Scanner;

public class q17\_lastIndexOf {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.err.println("Enter string : ");

String str = sc.nextLine();

System.out.println("Enter sub string : ");

String subStr = sc.next();

int x = str.lastIndexOf(subStr);

if (x == -1) {

System.out.printf("%s is not in %s\n", subStr, str);

} else {

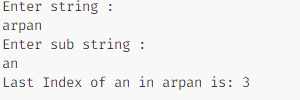
System.out.printf("Last Index of %s in %s is: %d\n", subStr, str, x);

}

sc.close();

}

}



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

import java.util.Scanner;

public class q18\_remove\_character {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Enter the character : ");

String ch = sc.next();

String replacedString = str.replace(ch, ""); // Replace with a space

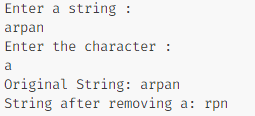
System.out.println("Original String: " + str);

System.out.println("String after removing " + ch + ": " + replacedString);

sc.close();

}

}



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

import java.util.Scanner;

public class q19\_replace\_substring {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Enter the string to be replaced and to replace : ");

String subStrToBeReplaced = sc.next();

String subStrToReplace = sc.next();

String newStr = str.replace(subStrToBeReplaced, subStrToReplace);

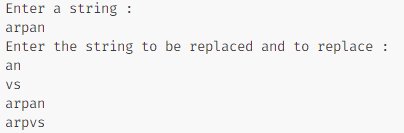
System.out.println(str);

System.out.println(newStr);

sc.close();

}

}



**Write a Java Program to Reverse a String.**

import java.util.Scanner;

public class q20\_reverse\_string {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

String newStr = "";

for (int i = str.length() - 1; i >= 0; i--) {

newStr += str.charAt(i);

}

System.out.println(newStr);

sc.close();

}

}



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

import java.util.Scanner;

public class q21\_search\_word {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Enter the word : ");

String word = sc.next();

int index = str.indexOf(word);

if (index != -1)

System.out.println("Found at " + index);

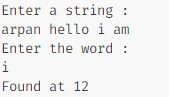
else

System.out.println("Not found");

sc.close();

}

}



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

import java.util.Scanner;

public class q22\_split\_substrings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

String[] substrings = str.split("\\s");

System.out.println("Substrings:");

for (String substring : substrings) {

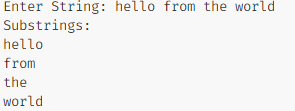
System.out.println(substring);

}

sc.close();

}

}



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

import java.util.Scanner;

public class q23\_search\_a\_word {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

System.out.print("Enter Word: ");

String word = sc.next();

int index = str.indexOf(word);

if (index != -1) {

System.out.println("Found at: " + index);

} else {

System.out.println("Not found");

}

sc.close();

}

}



**Write a Java Program to Replace All Occurring of a String.**

import java.util.Scanner;

public class q24\_replace\_all\_occurings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

System.out.print("Enter word to replace: ");

String word = sc.next();

System.out.print("Enter word you want to replace to: ");

String newWord = sc.next();

String newStr = str.replace(word, newWord);

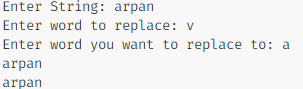
System.out.println(str);

System.out.println(newStr);

sc.close();

}

}



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

import java.util.Scanner;

public class q25\_uppercase {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

char[] charArray = str.toCharArray();

boolean isWordStart = true;

for (int i = 0; i < charArray.length; i++) {

if (Character.isLetter(charArray[i])) {

if (isWordStart) {

charArray[i] = Character.toUpperCase(charArray[i]);

isWordStart = false;

}

} else {

isWordStart = true;

}

}

String modifiedString = new String(charArray);

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

System.out.println("Modified String: " + modifiedString);

sc.close();

}

}



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

import java.util.HashSet;

import java.util.Scanner;

public class q26\_delete\_duplicates {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

String[] words = str.split("\\s");

HashSet<String> uniqueWords = new HashSet<>();

StringBuilder modifiedString = new StringBuilder();

for (String word : words) {

if (!uniqueWords.contains(word)) {

uniqueWords.add(word);

modifiedString.append(word + " ");

}

}

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

System.out.println("String after removing duplicates: " + modifiedString.toString().trim());

sc.close();

}

}



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

import java.util.Scanner;

public class q27\_reverse\_string {

public static String reverseIteration(String s) {

String newS = "";

for (int i = s.length() - 1; i >= 0; i--)

newS += s.charAt(i);

return newS;

}

public static String reverseRecursion(String s) {

if (s.isEmpty()) {

return s;

} else {

return reverseRecursion(s.substring(1)) + s.charAt(0);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

String strReverseIteration = reverseIteration(str);

String strReverseRecursion = reverseRecursion(str);

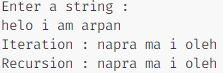
System.out.println("Iteration : " + strReverseIteration);

System.out.println("Recursion : " + strReverseRecursion);

sc.close();

}

}



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

import java.util.Scanner;

public class q28\_convert\_toUpperCase {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

str = str.toUpperCase();

System.out.println(str);

sc.close();

}

}



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

import java.util.HashSet;

import java.util.Scanner;

public class q29\_remove\_common\_characters {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter two Strings: ");

String str1 = sc.nextLine();

String str2 = sc.nextLine();

String result = removeChars(str1, str2);

System.out.println("Original String (First): " + str1);

System.out.println("Original String (Second): " + str2);

System.out.println("String after removing characters: " + result);

sc.close();

}

public static String removeChars(String str1, String str2) {

StringBuilder sb = new StringBuilder();

HashSet<Character> charSet = new HashSet<>();

for (char c : str1.toCharArray()) {

charSet.add(c);

}

for (char c : str2.toCharArray()) {

if (!charSet.contains(c)) {

sb.append(c);

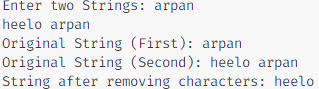
}

}

return sb.toString();

}

}



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

import java.util.Scanner;

public class q30\_consecutive\_vowel {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.println("Consecutive vowel substrings: " + findConsecutiveVowels(str));

sc.close();

}

public static String findConsecutiveVowels(String str) {

String vowels = "aeiouAEIOU";

int startIndex = -1;

char ch = ' ', c = ' ';

for (int i = 0; i < str.length() - 1; i++) {

ch = str.charAt(i);

c = str.charAt(i + 1);

if (vowels.indexOf(ch) != -1 && vowels.indexOf(c) != -1) {

startIndex = i;

break;

}

}

if (startIndex == -1) {

return "There are no Consecutive Vowels in this String";

} else {

return ch + " and " + c + " appear consecutively in this String";

}

}

}



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

import java.util.Scanner;

public class q31\_largest\_smallest\_word {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

findLargestSmallestWords(str);

sc.close();

}

public static void findLargestSmallestWords(String str) {

if (str.isEmpty()) {

System.out.println("String is empty.");

return;

}

String[] words = str.trim().split("\\s");

if (words.length == 1) {

System.out.println("String contains only one word: " + words[0]);

return;

}

String largestWord = words[0];

String smallestWord = words[0];

for (String word : words) {

if (word.length() > largestWord.length()) {

largestWord = word;

} else if (word.length() < smallestWord.length()) {

smallestWord = word;

}

}

System.out.println("Largest word: " + largestWord);

System.out.println("Smallest word: " + smallestWord);

}

}



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

import java.util.Scanner;

public class q32\_first\_last\_occurance {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

System.out.println("Enter the character : ");

String ch = sc.next();

int firstIndex = str.indexOf(ch);

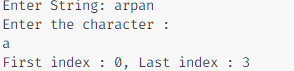
int lastIndex = str.lastIndexOf(ch);

System.out.println("First index : " + firstIndex + ", " + "Last index : " + lastIndex);

sc.close();

}

}



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

import java.util.Scanner;

public class q33\_display\_prime\_position\_characters {

public static boolean isPrime(int pos) {

boolean flag = false;

for (int i = 2; i < pos; i++) {

if (pos % i == 0) {

flag = true;

break;

}

}

if (flag) {

return false;

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

StringBuilder sb = new StringBuilder();

System.out.println("Characters at prime position : ");

for (int i = 2; i < str.length(); i++) {

if (isPrime(i))

sb.append(str.charAt(i)).append(" ");

}

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

sc.close();

}

}



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

import java.util.HashSet;

import java.util.Scanner;

import java.util.Arrays;

public class q34\_sort\_after\_deleting\_duplicates {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

String sortedStr = sortUnique(str);

System.out.println(sortedStr);

sc.close();

}

public static String sortUnique(String str) {

StringBuilder sb = new StringBuilder();

HashSet<Character> uniqueChars = new HashSet<>();

for (char ch : str.toCharArray()) {

if (ch != ' ' && !uniqueChars.contains(ch)) {

sb.append(ch);

uniqueChars.add(ch);

}

}

char[] charArray = sb.toString().toCharArray();

Arrays.sort(charArray);

return new String(charArray);

}

}



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

import java.util.\*;

class q35\_count\_replace {

static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

System.out.print("Enter String: ");

String s = sc.nextLine();

HashMap<Character, Integer> hmap = new HashMap<>();

for (char c : s.toCharArray()) {

hmap.put(c, hmap.getOrDefault(c, 0) + 1);

}

for (char c : hmap.keySet()) {

int x = s.indexOf(c);

s = s.substring(0, x) + hmap.get(c) + s.substring(x + 1);

}

System.out.println(s);

}

}



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

import java.util.Scanner;

public class q36\_last\_index\_of\_word {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

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

String word = sc.next();

int index = str.lastIndexOf(word);

System.out.println(index);

sc.close();

}

}



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

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class q37\_index\_of\_character {

public static List<Integer> findAllIndices(String text, String searchValue) {

List<Integer> indices = new ArrayList<>();

int startIndex = 0;

while (true) {

int index = text.indexOf(searchValue, startIndex);

if (index == -1) {

break;

}

indices.add(index);

startIndex = index + 1;

}

return indices;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a String: ");

String str = sc.nextLine();

System.out.println("Enter the character : ");

String ch = sc.next();

List<Integer> indices = findAllIndices(str, ch);

for (int index : indices) {

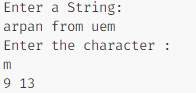
System.out.print(index + " ");

}

sc.close();

}

}



**Write a Java Program to Access the Characters or the ASCII of the Character available in the String.**

import java.util.Scanner;

public class q38\_print\_character\_ascii {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

for (int i = 0; i < str.length(); i++) {

char character = str.charAt(i);

int asciiValue = character;

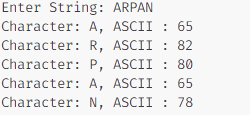
System.out.println("Character: " + character + ", ASCII : " + asciiValue);

}

sc.close();

}

}



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

import java.util.Scanner;

public class q39\_print\_character\_ascii {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter String: ");

String str = sc.nextLine();

for (int i = 0; i < str.length(); i++) {

char character = str.charAt(i);

int asciiValue = character;

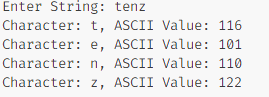
System.out.println("Character: " + character + ", ASCII Value: " + asciiValue);

}

sc.close();

}

}



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

import java.util.Scanner;

public class q40\_is\_first\_string\_a\_subset\_of\_second\_string {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

String str1 = scanner.nextLine();

System.out.print("Enter the second string: ");

String str2 = scanner.nextLine();

for (int i = 0; i < str1.length(); i++) {

char currentChar = str1.charAt(i);

if (str2.indexOf(currentChar) == -1) {

System.out.println("String 1 is not a subset of String 2");

System.exit(0);

}

}

System.out.println("String 1 is a subset of String 2");

scanner.close();

}

}



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

import java.util.Scanner;

public class q41\_frequency\_and\_position {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

System.out.print("Enter a character to search for: ");

char ch = sc.next().charAt(0);

int freq = 0;

System.out.print("Position(s) of occurrence: ");

for (int i = 0; i < str.length(); i++) {

if (str.charAt(i) == ch) {

freq++;

System.out.print((i + 1) + " ");

}

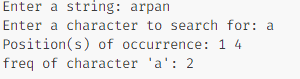
}

System.out.println("\nfreq of character '" + ch + "': " + freq);

sc.close();

}

}



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

import java.util.Scanner;

import java.util.HashMap;

public class q42\_frequency\_of\_characters {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String inputString = sc.nextLine();

inputString = inputString.toLowerCase();

System.out.println("Occurrences of each character (ignoring case):");

HashMap<Character, Integer> hm = new HashMap<>();

for (char c : inputString.toCharArray()) {

hm.put(c, hm.getOrDefault(c, 0) + 1);

}

System.out.println(hm);

sc.close();

}

}



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

import java.util.\*;

public class q43\_conversion\_from\_one\_string\_to\_another {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter two strings : ");

String src = sc.nextLine();

String dest = sc.nextLine();

int[] arr = new int[26];

for (int i = 0; i < src.length(); i++) {

arr[src.charAt(i) - 'a']--;

}

for (int i = 0; i < dest.length(); i++) {

arr[dest.charAt(i) - 'a']++;

}

int sum = 0;

for (int i = 0; i < 26; i++) {

sum += Math.abs(arr[i]);

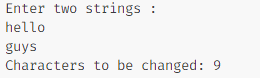
}

System.out.println("Characters to be changed: " + sum);

sc.close();

}

}



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

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class q44\_date {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a date (in format yyyy-MM-dd): ");

String inputDate = scanner.nextLine();

if (isValidDateFormat(inputDate, "yyyy-MM-dd")) {

System.out.println("The date is in proper format.");

} else {

System.out.println("The date is not in proper format.");

}

scanner.close();

}

public static boolean isValidDateFormat(String dateStr, String format) {

SimpleDateFormat sdf = new SimpleDateFormat(format);

sdf.setLenient(false);

try {

Date date = sdf.parse(dateStr);

return date != null;

} catch (ParseException e) {

return false;

}

}

}



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

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class q45\_valid\_email {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an email address: ");

String email = scanner.nextLine();

if (isValidEmail(email)) {

System.out.println("The email address is valid.");

} else {

System.out.println("The email address is not valid.");

}

scanner.close();

}

public static boolean isValidEmail(String email) {

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);

return matcher.matches();

}

}



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

public class q46\_string\_buffer {

public static void main(String[] args) {

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append("Hello");

stringBuffer.append(" ");

stringBuffer.append("world");

System.out.println("Content of StringBuffer: " + stringBuffer);

stringBuffer.append(", ");

stringBuffer.append("how");

stringBuffer.append(" ");

stringBuffer.append("are");

stringBuffer.append(" ");

stringBuffer.append("you");

System.out.println("Updated content of StringBuffer: " + stringBuffer);

stringBuffer.insert(12, "beautiful ");

System.out.println("Content after insertion: " + stringBuffer);

stringBuffer.replace(21, 24, "doing");

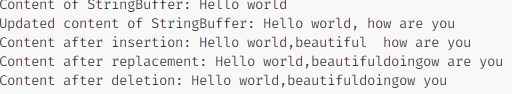
System.out.println("Content after replacement: " + stringBuffer);

stringBuffer.delete(29, 33);

System.out.println("Content after deletion: " + stringBuffer);

}

}



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

import java.lang.reflect.Method;

import java.util.Scanner;

public class q47\_verify\_a\_card {

public static void main(String[] args) {

Class<StringBuffer> stringBufferClass = StringBuffer.class;

Method[] stringBufferMethods = stringBufferClass.getDeclaredMethods();

Scanner sc = new Scanner(System.in);

System.out.print("Enter a Method Name: ");

String methodName = sc.nextLine();

boolean methodExists = false;

for (Method method : stringBufferMethods) {

if (method.getName().equals(methodName)) {

methodExists = true;

break;

}

}

if (methodExists) {

System.out.println("The method '" + methodName + "' exists in the StringBuffer class.");

} else {

System.out.println("The method '" + methodName + "' does not exist in the StringBuffer class.");

}

sc.close();

}

}



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

import java.util.Scanner;

public class q48\_print\_name {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter your name : ");

String name = sc.nextLine();

System.out.printf("Hello %s, nice to meet you!\n", name);

sc.close();

}

}



**Write a Java Program to Count a Group of Words in a String.**

import java.util.Scanner;

public class q49\_count\_group\_of\_words {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String input = scanner.nextLine();

System.out.print("Enter group of words you want to count: ");

String group = scanner.nextLine();

int start = input.indexOf(group);

int count = 0;

while (start != -1) {

count++;

input = input.substring(start + 1);

start = input.indexOf(group);

}

System.out.println("This group of words has appeared " + count + " times in input string");

scanner.close();

}

}



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

import java.util.Scanner;

public class q50\_count\_words {

public static int wordCount(String s) {

String[] words = s.trim().split("\\s");

return words.length;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a text : ");

String text = sc.nextLine();

int count = wordCount(text);

System.out.printf("Number of word(s) in the string : %d\n", count);

sc.close();

}

}



WEEK - 9

**Write a Java program in which total 4 threads should run. Set different priorities to the thread.**

public class q1\_4\_threads\_should\_run {

public static void main(String[] args) {

Thread t1 = new Thread(new MyRunnable(), "Thread 1");

Thread t2 = new Thread(new MyRunnable(), "Thread 2");

Thread t3 = new Thread(new MyRunnable(), "Thread 3");

Thread t4 = new Thread(new MyRunnable(), "Thread 4");

t1.setPriority(Thread.MIN\_PRIORITY); // Priority 1

t2.setPriority(Thread.NORM\_PRIORITY); // Priority 5

t3.setPriority(Thread.NORM\_PRIORITY); // Priority 5

t4.setPriority(Thread.MAX\_PRIORITY); // Priority 10

t1.start();

t2.start();

t3.start();

t4.start();

}

static class MyRunnable implements Runnable {

public void run() {

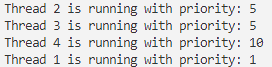
System.out.println(Thread.currentThread().getName() + " is running with priority: "

+ Thread.currentThread().getPriority());

}

}

}



**Create 4 threads with priority 1,3,5,7 respectively. Update a counter in each of the threads for 10 ms. Print the final value of count for each thread.**

public class q2\_counter {

static int counter = 0;

public static void main(String[] args) {

Thread t1 = new Thread(new MyRunnable(1));

Thread t2 = new Thread(new MyRunnable(3));

Thread t3 = new Thread(new MyRunnable(5));

Thread t4 = new Thread(new MyRunnable(7));

t1.start();

t2.start();

t3.start();

t4.start();

try {

t1.join();

t2.join();

t3.join();

t4.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final count for Thread 1: " + MyRunnable.count1);

System.out.println("Final count for Thread 2: " + MyRunnable.count2);

System.out.println("Final count for Thread 3: " + MyRunnable.count3);

System.out.println("Final count for Thread 4: " + MyRunnable.count4);

}

static class MyRunnable implements Runnable {

static int count1 = 0, count2 = 0, count3 = 0, count4 = 0;

int priority;

MyRunnable(int priority) {

this.priority = priority;

}

public void run() {

Thread.currentThread().setPriority(priority);

for (int i = 0; i < 10; i++) {

synchronized (this) {

counter++;

switch (priority) {

case 1:

count1++;

break;

case 3:

count2++;

break;

case 5:

count3++;

break;

case 7:

count4++;

break;

}

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

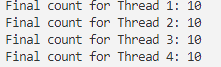
}

}

}

}

}



**Write a Java Program to Use Method Level Synchronization.**

public class q3\_Method\_Level\_Synchronization {

private static int counter = 0;

public static void main(String[] args) {

Thread t1 = new Thread(new MyRunnable(), "Thread 1");

Thread t2 = new Thread(new MyRunnable(), "Thread 2");

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final count: " + counter);

}

static class MyRunnable implements Runnable {

public void run() {

for (int i = 0; i < 1000; i++) {

incrementCounter();

}

}

private synchronized void incrementCounter() {

counter++;

}

}

}



**Write a Java Program to Use Block Level Synchronization.**

public class q4\_Block\_Level\_Synchronization {

private static int counter = 0;

private static final Object lock = new Object();

public static void main(String[] args) {

Thread t1 = new Thread(new MyRunnable(), "Thread 1");

Thread t2 = new Thread(new MyRunnable(), "Thread 2");

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final count: " + counter);

}

static class MyRunnable implements Runnable {

public void run() {

for (int i = 0; i < 1000; i++) {

incrementCounter();

}

}

private void incrementCounter() {

synchronized (lock) {

counter++;

}

}

}

}



**Write a Java Program to Check Whether Define run() Method as Synchronized.**

public class q5\_Check\_Whether\_Define\_run\_Method\_as\_Synchronized {

private static int counter = 0;

public static void main(String[] args) {

Thread t1 = new Thread(new MyRunnable(), "Thread 1");

Thread t2 = new Thread(new MyRunnable(), "Thread 2");

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final count: " + counter);

}

static class MyRunnable implements Runnable {

public synchronized void run() {

for (int i = 0; i < 1000; i++) {

incrementCounter();

}

}

private void incrementCounter() {

counter++;

}

}

}



**Write a Java Program to Solve Producer Consumer Problem Using Synchronization.**

import java.util.LinkedList;

public class q6\_Producer\_Consumer\_Problem\_Using\_Synchronization {

public static void main(String[] args) {

Buffer buffer = new Buffer();

new Thread(() -> buffer.produce()).start();

new Thread(() -> buffer.consume()).start();

}

}

class Buffer {

private final LinkedList<Integer> queue = new LinkedList<>();

private final int capacity = 5;

public synchronized void produce() {

try {

for (int i = 0; i < 10; i++) {

while (queue.size() == capacity) wait();

queue.add(i);

System.out.println("Produced: " + i);

notify();

Thread.sleep(1000);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public synchronized void consume() {

try {

for (int i = 0; i < 10; i++) {

while (queue.isEmpty()) wait();

int consumed = queue.removeFirst();

System.out.println("Consumed: " + consumed);

notify();

Thread.sleep(2000);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**   **

**Write a Java Program to Show that Method Will be Verified whether it is Synchronized or Not.**

import java.lang.reflect.Method;

import java.lang.reflect.Modifier;

public class q7\_Show\_that\_Method\_Will\_be\_Verified\_Whether\_it\_is\_Synchronized\_or\_Not {

public static void main(String[] args) {

Method[] methods = SynchronizedClass.class.getDeclaredMethods();

for (Method method : methods) {

System.out.println("Method: " + method.getName());

boolean isSynchronized = isMethodSynchronized(method);

System.out.println("Synchronized: " + (isSynchronized ? "Yes" : "No"));

}

}

private static boolean isMethodSynchronized(Method method) {

return (method.getModifiers() & Modifier.SYNCHRONIZED) != 0;

}

}

class SynchronizedClass {

public synchronized void synchronizedMethod() {

System.out.println("Synchronized method is being executed.");

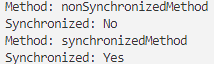
}

public void nonSynchronizedMethod() {

System.out.println("Non-Synchronized method is being executed.");

}

}

****

**Write a Java Program to Show How Can Class Object be Locked Using Method Level Synchronization.**

public class q8\_Show\_How\_Can\_Class\_Object\_be\_Locked\_Using\_Method\_Level\_Synchronization {

public static void main(String[] args) {

Thread thread1 = new Thread(new MyRunnable(), "Thread 1");

Thread thread2 = new Thread(new MyRunnable(), "Thread 2");

thread1.start();

thread2.start();

}

}

class MyClass {

public synchronized void synchronizedMethod() {

System.out.println(Thread.currentThread().getName() + " is executing synchronized method.");

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

class MyRunnable implements Runnable {

private static MyClass myClass = new MyClass();

public void run() {

myClass.synchronizedMethod();

}

}

****

**Write a Java Program to Synchronize the Threads Acting on the Same Object. The Synchronized Block in the Program can be executed by Only One Thread at a Time.**

public class q9\_Synchronize\_the\_Threads\_Acting\_on\_the\_Same\_Object {

public static void main(String[] args) {

SharedObject sharedObject = new SharedObject();

Thread thread1 = new Thread(new MyRunnable(sharedObject), "Thread 1");

Thread thread2 = new Thread(new MyRunnable(sharedObject), "Thread 2");

thread1.start();

thread2.start();

}

}

class SharedObject {

public void synchronizedMethod() {

synchronized (this) {

System.out.println(Thread.currentThread().getName() + " is executing synchronized method.");

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

class MyRunnable implements Runnable {

private SharedObject sharedObject;

public MyRunnable(SharedObject sharedObject) {

this.sharedObject = sharedObject;

}

public void run() {

sharedObject.synchronizedMethod();

}

}



**Write a Java Program to Avoid Dead Locks.**

public class q10\_string\_compare {

private int balance = 10000;

public void deposit(int amount) {

synchronized (this) {

balance += amount;

}

}

public void withdraw(int amount) {

synchronized (this) {

balance -= amount;

}

}

public int getBalance() {

return balance;

}

public void transfer(q10\_string\_compare target, int amount) {

if (this == target) {

System.out.println("Cannot transfer to the same account.");

return;

}

q10\_string\_compare first = this;

q10\_string\_compare second = target;

if (System.identityHashCode(this) > System.identityHashCode(target)) {

first = target;

second = this;

}

synchronized (first) {

synchronized (second) {

if (amount > balance) {

System.out.println("Insufficient funds for transfer.");

return;

}

this.withdraw(amount);

target.deposit(amount);

}

}

}

}

public class q10\_2nd\_Avoid\_Dead\_Locks {

public static void main(String[] args) {

final q10\_string\_compare acc1 = new q10\_string\_compare();

final q10\_string\_compare acc2 = new q10\_string\_compare();

Thread thread1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

acc1.transfer(acc2, 10);

}

});

Thread thread2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

acc2.transfer(acc1, 10);

}

});

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Balance in Account 1: " + acc1.getBalance());

System.out.println("Balance in Account 2: " + acc2.getBalance());

}

}

****

**Write a Java Program to Solve Deadlock Using Thread.**

public class q11\_Deadlock\_Using\_Thread {

private static final Object lock1 = new Object();

private static final Object lock2 = new Object();

public static void main(String[] args) {

Thread thread1 = new Thread(() -> {

synchronized (lock1) {

System.out.println("Thread 1: Holding lock 1...");

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Thread 1: Waiting for lock 2...");

synchronized (lock2) {

System.out.println("Thread 1: Holding lock 1 and lock 2...");

}

}

});

Thread thread2 = new Thread(() -> {

synchronized (lock1) {

System.out.println("Thread 2: Holding lock 1...");

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Thread 2: Waiting for lock 2...");

synchronized (lock2) {

System.out.println("Thread 2: Holding lock 1 and lock 2...");

}

}

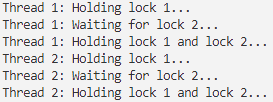
});

thread1.start();

thread2.start();

}

}

****

**Write a Java Program to Create a Thread that Implement the Runnable Interface.**

public class q12\_Thread\_that\_Implement\_the\_Runnable\_Interface implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

public static void main(String[] args) {

q12\_Thread\_that\_Implement\_the\_Runnable\_Interface myRunnable = new q12\_Thread\_that\_Implement\_the\_Runnable\_Interface();

Thread thread = new Thread(myRunnable);

thread.start();

}

}

****

**Write a Java Program to Show the Priority in Threads.**

public class q13\_the\_Priority\_in\_Threads {

public static void main(String[] args) {

Thread thread1 = new Thread(new MyRunnable(), "Thread 1");

Thread thread2 = new Thread(new MyRunnable(), "Thread 2");

Thread thread3 = new Thread(new MyRunnable(), "Thread 3");

thread1.setPriority(Thread.MIN\_PRIORITY);

thread2.setPriority(Thread.NORM\_PRIORITY);

thread3.setPriority(Thread.MAX\_PRIORITY);

thread1.start();

thread2.start();

thread3.start();

System.out.println(thread1.getName() + " priority: " + thread1.getPriority());

System.out.println(thread2.getName() + " priority: " + thread2.getPriority());

System.out.println(thread3.getName() + " priority: " + thread3.getPriority());

}

static class MyRunnable implements Runnable {

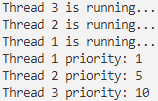
public void run() {

System.out.println(Thread.currentThread().getName() + " is running...");

}

}

}

****

**Write a Java Program to Check Priority Level of a Thread.**

public class q14\_Check\_Priority\_Level\_of\_a\_Thread {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

int priority = thread.getPriority();

System.out.println("Thread priority: " + priority);

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running...");

}

}

}

****

**Write a Java Program to Set the Priority of a Thread.**

public class q15\_Set\_the\_Priority\_of\_a\_Thread {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

thread.setPriority(Thread.MAX\_PRIORITY);

thread.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running with priority: " + Thread.currentThread().getPriority());

}

}

}

****

**Write a Java Program to Get the Priorities of Running Threads.**

public class q16\_Get\_the\_Priorities\_of\_Running\_Threads {

public static void main(String[] args) {

Thread thread1 = new Thread(new MyRunnable(), "Thread 1");

Thread thread2 = new Thread(new MyRunnable(), "Thread 2");

thread1.start();

thread2.start();

int priority1 = thread1.getPriority();

int priority2 = thread2.getPriority();

System.out.println(thread1.getName() + " priority: " + priority1);

System.out.println(thread2.getName() + " priority: " + priority2);

}

static class MyRunnable implements Runnable {

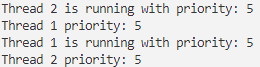
public void run() {

System.out.println(Thread.currentThread().getName() + " is running with priority: " + Thread.currentThread().getPriority());

}

}

}

****

**Write a Java Program to Access the Priority You Can Use Method with Thread Object.**

public class q17\_Access\_the\_Priority\_You\_Can\_Use\_Method\_With\_Thread\_Object {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

int priority = thread.getPriority();

System.out.println("Thread priority before setting: " + priority);

thread.setPriority(Thread.MAX\_PRIORITY);

int updatedPriority = thread.getPriority();

System.out.println("Thread priority after setting: " + updatedPriority);

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running with priority: " + Thread.currentThread().getPriority());

}

}

}

****

**Write a Java Program to Use Join Thread.**

public class q18\_Use\_Join\_Thread {

public static void main(String[] args) {

Thread thread1 = new Thread(new MyRunnable(), "Thread 1");

Thread thread2 = new Thread(new MyRunnable(), "Thread 2");

thread1.start();

try {

thread1.join();

System.out.println("Thread 1 has finished.");

} catch (InterruptedException e) {

e.printStackTrace();

}

thread2.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running...");

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

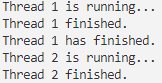
}

System.out.println(Thread.currentThread().getName() + " finished.");

}

}

}

****

**Write a Java Program Defining Thread By Extending Thread.**

public class q19\_Defining\_Thread\_By\_Extending\_Thread extends Thread {

public void run() {

System.out.println("Thread is running...");

}

public static void main(String[] args) {

q19\_Defining\_Thread\_By\_Extending\_Thread thread = new q19\_Defining\_Thread\_By\_Extending\_Thread();

thread.start();

}

}

****

**Write a Java Program to Handle IllegalThreadStateException.**

public class q20\_Handle\_IllegalThreadStateException {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

thread.start();

try {

thread.start();

} catch (IllegalThreadStateException e) {

System.out.println("IllegalThreadStateException caught: " + e.getMessage());

}

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running...");

}

}

}

****

**Write a Java Program to Check Whether Static Block will be used.**

public class q21\_Check\_Whether\_Static\_Block\_will\_be\_Used {

static {

System.out.println("Static block is executed.");

}

public static void main(String[] args) {

System.out.println("Main method is executed.");

}

}

****

**Write a Java Program to Show Why Exit Method is used in Static Method.**

public class q22\_Show\_Why\_Exit\_Method\_is\_Used\_in\_Static\_Method {

public static void main(String[] args) {

System.out.println("Main method is executing...");

myStaticMethod();

}

public static void myStaticMethod() {

System.out.println("Static method is executing...");

System.exit(0);

}

}

****

**Write a Java Program to Illustrate Thread Example for setName(string name).**

public class q23\_Illustrate\_Thread\_Example\_for\_setName {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.setName("MyThread");

thread.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread name: " + Thread.currentThread().getName());

}

}

}

****

**Write a Java Program to Illustrate Thread Example for Destroy().**

class MyThread extends Thread {

private volatile boolean running = true;

@Override

public void run() {

while (running) {

try {

System.out.println("Thread is running...");

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Thread interrupted.");

Thread.currentThread().interrupt();

}

}

System.out.println("Thread stopped.");

}

public void stopThread() {

running = false;

}

}

public class q24\_Illustrate\_Thread\_Example\_for\_Destroy {

public static void main(String[] args) {

MyThread thread = new MyThread();

thread.start();

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

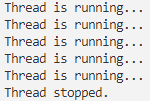
e.printStackTrace();

}

thread.stopThread();

}

}



**Write a Java Program to Illustrate Thread Example for suspend().**

class MyThread extends Thread {

private boolean suspended = false;

public void suspendThread() {

suspended = true;

}

public void resumeThread() {

suspended = false;

synchronized (this) {

notify();

}

}

@Override

public void run() {

while (true) {

try {

while (suspended) {

synchronized (this) {

wait();

}

}

System.out.println("Thread is running...");

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Thread interrupted.");

Thread.currentThread().interrupt();

}

}

}

}

public class q25\_Illustrate\_Thread\_Example\_for\_suspend {

public static void main(String[] args) {

MyThread thread = new MyThread();

thread.start();

try {

Thread.sleep(5000);

thread.suspendThread();

System.out.println("Thread suspended for 3 seconds...");

Thread.sleep(3000);

thread.resumeThread();

System.out.println("Thread resumed...");

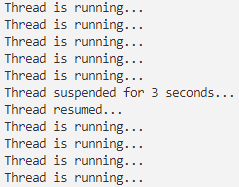
} catch (InterruptedException e) {

e.printStackTrace();

}

}

}



**Write a Java Program to Illustrate Thread Example for currentThread().**

public class q26\_Illustrate\_Thread\_Example\_for\_currentThread {

public static void main(String[] args) {

Thread thread = Thread.currentThread();

System.out.println("Current thread: " + thread.getName());

}

}

****

**Write a Java Program to Illustrate Thread Example for run().**

public class q27\_Illustrate\_Thread\_Example\_for\_run {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Illustrate Thread Example for getThreadGroup().**

public class q28\_Illustrate\_Thread\_Example\_for\_getThreadGroup {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

System.out.println("Thread group: " + thread.getThreadGroup().getName());

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Illustrate Thread Example for getPriority().**

public class q29\_Illustrate\_Thread\_Example\_for\_getPriority {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

int priority = thread.getPriority();

System.out.println("Thread priority: " + priority);

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Illustrate Thread Example for Alive().**

public class q30\_Illustrate\_Thread\_Example\_for\_Alive {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

System.out.println("Thread is alive before starting: " + thread.isAlive());

thread.start();

System.out.println("Thread is alive after starting: " + thread.isAlive());

}

static class MyRunnable implements Runnable {

@Override

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Illustrate Thread Example for getName().**

public class q31\_Illustrate\_Thread\_Example\_for\_getName {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

System.out.println("Thread name: " + thread.getName());

}

static class MyRunnable implements Runnable {

@Override

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Show Interfaces Can be Extended.**

interface MyInterface {

void display();

}

interface MyExtendedInterface extends MyInterface {

void show();

}

public class q32\_Show\_Interfaces\_Can\_be\_Extended implements MyExtendedInterface {

public void display() {

System.out.println("Display method implementation");

}

public void show() {

System.out.println("Show method implementation");

}

public static void main(String[] args) {

q32\_Show\_Interfaces\_Can\_be\_Extended example = new q32\_Show\_Interfaces\_Can\_be\_Extended();

example.display();

example.show();

}

}

****

**Write a Java Program to Check a Thread is Alive or Not.**

interface MyInterface {

void display();

}

interface MyExtendedInterface extends MyInterface {

void show();

}

public class q33\_Check\_a\_Thread\_is\_Alive\_or\_Not implements MyExtendedInterface {

public void display() {

System.out.println("Display method implementation");

}

public void show() {

System.out.println("Show method implementation");

}

public static void main(String[] args) {

q33\_Check\_a\_Thread\_is\_Alive\_or\_Not example = new q33\_Check\_a\_Thread\_is\_Alive\_or\_Not();

example.display();

example.show();

}

}

****

**Write a Java Program to Get the Name of a Running Thread.**

public class q34\_Get\_the\_Name\_of\_a\_Running\_Thread {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

thread.start();

System.out.println("Running thread name: " + Thread.currentThread().getName());

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Get the Name of the Thread.**

public class q35\_Get\_the\_Name\_of\_the\_Thread {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable(), "MyThread");

System.out.println("Thread name: " + thread.getName());

}

static class MyRunnable implements Runnable {

@Override

public void run() {

System.out.println("Thread is running...");

}

}

}

****

**Write a Java Program to Check if a Given run() Method is Overloaded in the Thread Class.**

public class q36\_Check\_if\_a\_Given\_run\_Method\_is\_Overloaded\_in\_the\_Thread\_Class {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("run() method is overloaded.");

}

public void run(String message) {

System.out.println("run() method is overloaded with message: " + message);

}

}

}

****

**Write a Java Program to Check Whether Define a Thread Class without Defining**

**run() Method in the Class.**

public class q37\_Check\_Whether\_Define\_a\_Thread\_Class\_Without\_Defining\_run\_Method\_in\_the\_Class {

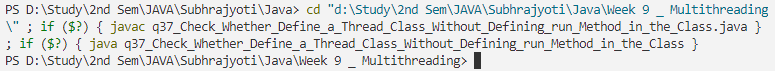
public static void main(String[] args) {

Thread thread = new Thread();

thread.start();

}

}



**Write a Java Program to Stop a Thread.**

public class q38\_Stop\_a\_Thread {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.start();

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

thread.interrupt();

}

static class MyRunnable implements Runnable {

public void run() {

while (!Thread.currentThread().isInterrupted()) {

System.out.println("Thread is running...");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

Thread.currentThread().interrupt(); // Reset the interrupted status

}

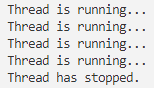
}

System.out.println("Thread has stopped.");

}

}

}

****

**Write a Java Program to Suspend a Thread for a While.**

public class q39\_Suspend\_a\_Thread\_for\_a\_While {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.start();

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

try {

Thread.sleep(2000); // Suspend the thread for 2 seconds

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Thread resumes...");

}

}

}

****

**Write a Java Program to Check a Thread has Stopped or Not.**

public class q40\_Check\_a\_Thread\_has\_Stopped\_or\_Not {

public static void main(String[] args) {

Thread thread = new Thread(new MyRunnable());

thread.start();

try {

Thread.sleep(2000); // Sleep for 2 seconds

} catch (InterruptedException e) {

e.printStackTrace();

}

if (!thread.isAlive()) {

System.out.println("Thread has stopped.");

} else {

System.out.println("Thread is still running.");

}

}

static class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

}

}

****