PROGRAM QUESTIONS (16 JULY)

1)Wrie a Java progam to Create a class Known as "Bank accoent" with methods called deposit () and Withdraw(). Create a subclass called saving account that withdrawn () method to prevent withdrawals if the account balance falls below one hundred.

class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

if (initialBalance >= 0) {

balance = initialBalance;

} else {

balance = 0;

System.out.println("Initial balance can't be negative. Setting balance to 0.");

}

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Deposit amount must be positive.");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrew: " + amount);

} else {

System.out.println("Withdrawal amount must be positive and within available balance.");

}

}

public double getBalance() {

return balance;

}

}

class SavingAccount extends BankAccount {

public SavingAccount(double initialBalance) {

super(initialBalance);

}

@Override

public void withdraw(double amount) {

if (getBalance() - amount >= 100) {

super.withdraw(amount);

} else {

System.out.println("Cannot withdraw: Balance would fall below $100.");

}

}

}

public class Main {

public static void main(String[] args) {

SavingAccount myAccount = new SavingAccount(500);

myAccount.deposit(200);

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

myAccount.withdraw(150);

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

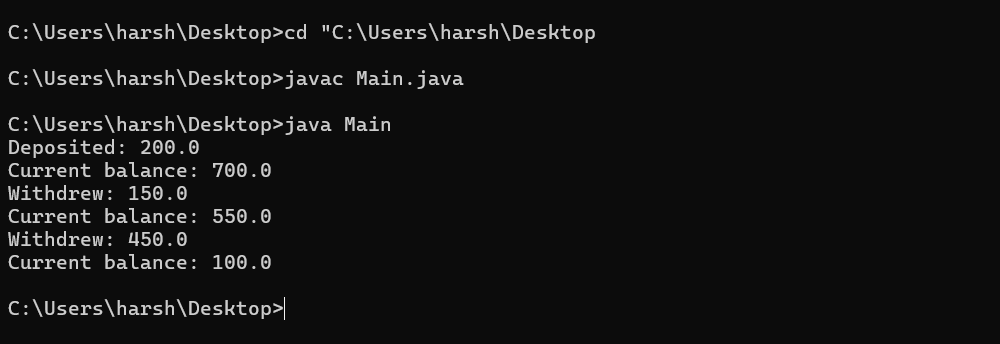
myAccount.withdraw(450);

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

}

}

Output:



2) Write a Java program to create a base class Known as Person with methods called get First Name land git Last Name () .

Create a subclas called Employer that adds a new method named get Employeed C) and overides the get last Name method to indude the employee's job tittle.

class Person {

public String getFirstName() {

return "John";

}

public String getLastName() {

return "Doe";

}

}

class Employee extends Person {

public void getEmployed() {

System.out.println("Employee has been hired.");

}

@Override

public String getLastName() {

return "Smith";

}

public double calculateArea(double radius) {

return Math.PI \* radius \* radius;

}

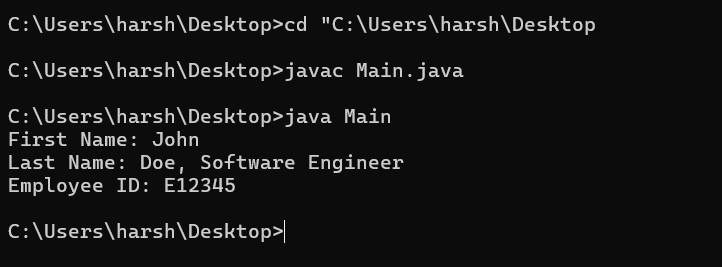
public double calculatePerimeter(double radius) {

return 2 \* Math.PI \* radius;

}

}

Output:



3) Write a Java grogram 70 Create a class Called shape with Methods called get Perimeter () and get Area (). Create a Subclass called O that overrides the get Perimeter () and get Area () methods to calculate the area and perimeter the circle.

abstract class Shape {

public abstract double getPerimeter();

public abstract double getArea();

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

@Override

public double getArea() {

return Math.PI \* radius \* radius;

}

}

public class Main {

public static void main(String[] args) {

Circle circle = new Circle(5);

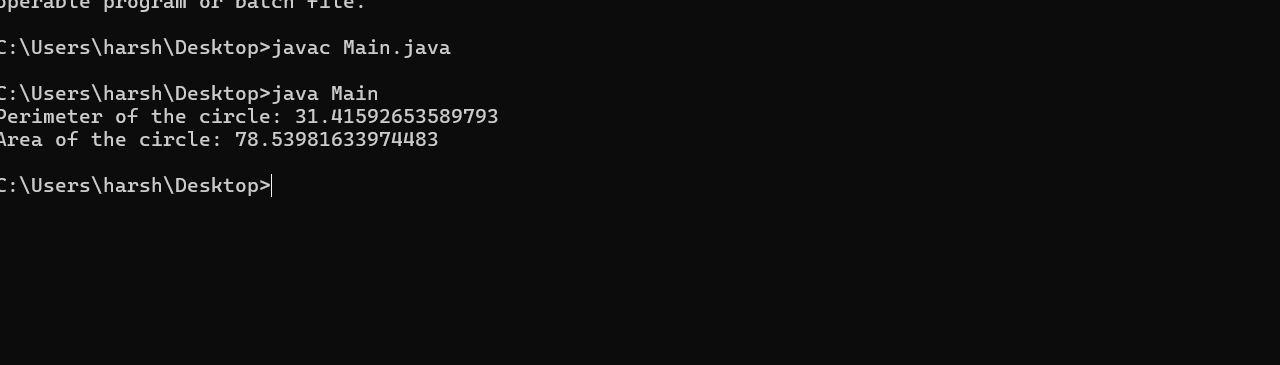
System.out.println("Perimeter of the circle: " + circle.getPerimeter());

System.out.println("Area of the circle: " + circle.getArea());

}

}

Output:



4) Write a Java program to create a Vehicles class heirachy, the base class should be Vehicle, which subclasses Truck, Car and Motorcycle. Each Subclass should have properties such as make, model, year and fueltype Implement methods for calculating fuel efficienceny, distance traveled and maximum speed.

abstract class Vehicle {

private String make;

private String model;

private int year;

private String fuelType;

public Vehicle(String make, String model, int year, String fuelType) {

this.make = make;

this.model = model;

this.year = year;

this.fuelType = fuelType;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public int getYear() {

return year;

}

public String getFuelType() {

return fuelType;

}

public abstract double calculateFuelEfficiency();

public abstract double calculateDistanceTraveled(double fuelUsed);

public abstract double getMinimumSpeed();

@Override

public String toString() {

return year + " " + make + " " + model + " (" + fuelType + ")";

}

}

class Truck extends Vehicle {

public Truck(String make, String model, int year, String fuelType) {

super(make, model, year, fuelType);

}

@Override

public double calculateFuelEfficiency() {

return 15.0;

}

@Override

public double calculateDistanceTraveled(double fuelUsed) {

return fuelUsed \* calculateFuelEfficiency();

}

@Override

public double getMinimumSpeed() {

return 40.0;

}

}

class Car extends Vehicle {

public Car(String make, String model, int year, String fuelType) {

super(make, model, year, fuelType);

}

@Override

public double calculateFuelEfficiency() {

return 25.0;

}

@Override

public double calculateDistanceTraveled(double fuelUsed) {

return fuelUsed \* calculateFuelEfficiency();

}

@Override

public double getMinimumSpeed()

return 30.0;

}

}

class Motorcycle extends Vehicle {

public Motorcycle(String make, String model, int year, String fuelType) {

super(make, model, year, fuelType);

}

@Override

public double calculateFuelEfficiency() {

return 50.0;

}

@Override

public double calculateDistanceTraveled(double fuelUsed) {

return fuelUsed \* calculateFuelEfficiency();

}

@Override

public double getMinimumSpeed() {

return 20.0;

}

}

public class Main {

public static void main(String[] args) {

Vehicle truck = new Truck("Ford", "F-150", 2020, "Diesel");

Vehicle car = new Car("Toyota", "Camry", 2021, "Gasoline");

Vehicle motorcycle = new Motorcycle("Harley-Davidson", "Sportster", 2019, "Gasoline");

System.out.println(truck);

System.out.println("Fuel Efficiency: " + truck.calculateFuelEfficiency() + " mpg");

System.out.println("Distance Traveled with 10 gallons: " + truck.calculateDistanceTraveled(10) + " miles");

System.out.println("Minimum Speed: " + truck.getMinimumSpeed() + " mph");

System.out.println();

System.out.println(car);

System.out.println("Fuel Efficiency: " + car.calculateFuelEfficiency() + " mpg");

System.out.println("Distance Traveled with 10 gallons: " + car.calculateDistanceTraveled(10) + " miles");

System.out.println("Minimum Speed: " + car.getMinimumSpeed() + " mph");

System.out.println();

System.out.println(motorcycle);

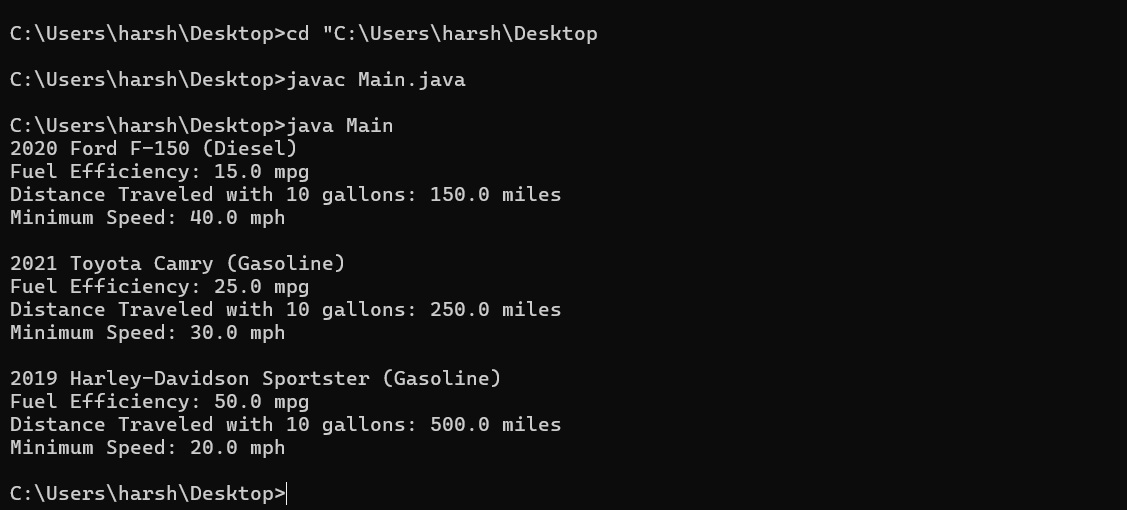
System.out.println("Fuel Efficiency: " + motorcycle.calculateFuelEfficiency() + " mpg");

System.out.println("Distance Traveled with 10 gallons: " + motorcycle.calculateDistanceTraveled(10) + " miles");

System.out.println("Minimum Speed: " + motorcycle.getMinimumSpeed() + " mph");

}

}

Output:

5) Write a Java system program that Creates a class history hierarchy for employeess of a company. The base should be employeee with subclass manager, developer and programmer Each subclass should have properties such as name, address, salary and job tittle.Implement methods for calculating bonuses, generating performance reports and manging projects.

**abstract class Employee {**

**private String name;**

**private String address;**

**private double salary;**

**private String jobTitle;**

**public Employee(String name, String address, double salary, String jobTitle) {**

**this.name = name;**

**this.address = address;**

**this.salary = salary;**

**this.jobTitle = jobTitle;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public double getSalary() {**

**return salary;**

**}**

**public String getJobTitle() {**

**return jobTitle;**

**}**

**public abstract double calculateBonus();**

**public abstract String generatePerformanceReport();**

**public abstract void manageProjects();**

**@Override**

**public String toString() {**

**return name + ", " + jobTitle + ", " + address + ", $" + salary;**

**}**

**}**

**class Manager extends Employee {**

**public Manager(String name, String address, double salary, String jobTitle) {**

**super(name, address, salary, jobTitle);**

**}**

**@Override**

**public double calculateBonus() {**

**return getSalary() \* 0.20;**

**}**

**@Override**

**public String generatePerformanceReport() {**

**return "Manager " + getName() + " has exceeded expectations in Q1.";**

**}**

**@Override**

**public void manageProjects() {**

**System.out.println(getName() + " is managing high-level projects.");**

**}**

**}**

**class Developer extends Employee {**

**public Developer(String name, String address, double salary, String jobTitle) {**

**super(name, address, salary, jobTitle);**

**}**

**@Override**

**public double calculateBonus() {**

**return getSalary() \* 0.15;**

**}**

**@Override**

**public String generatePerformanceReport() {**

**return "Developer " + getName() + " has delivered multiple successful projects in Q1.";**

**}**

**@Override**

**public void manageProjects() {**

**System.out.println(getName() + " is managing software development projects.");**

**}**

**}**

**class Programmer extends Employee {**

**public Programmer(String name, String address, double salary, String jobTitle) {**

**super(name, address, salary, jobTitle);**

**}**

**@Override**

**public double calculateBonus() {**

**return getSalary() \* 0.10;**

**}**

**@Override**

**public String generatePerformanceReport() {**

**return "Programmer " + getName() + " has written high-quality code and fixed critical bugs in Q1.";**

**}**

**@Override**

**public void manageProjects() {**

**System.out.println(getName() + " is working on coding tasks and bug fixes.");**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Employee manager = new Manager("Alice Johnson", "123 Main St", 120000, "Manager");**

**Employee developer = new Developer("Bob Smith", "456 Elm St", 90000, "Developer");**

**Employee programmer = new Programmer("Charlie Brown", "789 Oak St", 70000, "Programmer");**

**System.out.println(manager);**

**System.out.println("Bonus: $" + manager.calculateBonus());**

**System.out.println("Performance Report: " + manager.generatePerformanceReport());**

**manager.manageProjects();**

**System.out.println();**

**System.out.println(developer);**

**System.out.println("Bonus: $" + developer.calculateBonus());**

**System.out.println("Performance Report: " + developer.generatePerformanceReport());**

**developer.manageProjects();**

**System.out.println();**

**System.out.println(programmer);**

**System.out.println("Bonus: $" + programmer.calculateBonus());**

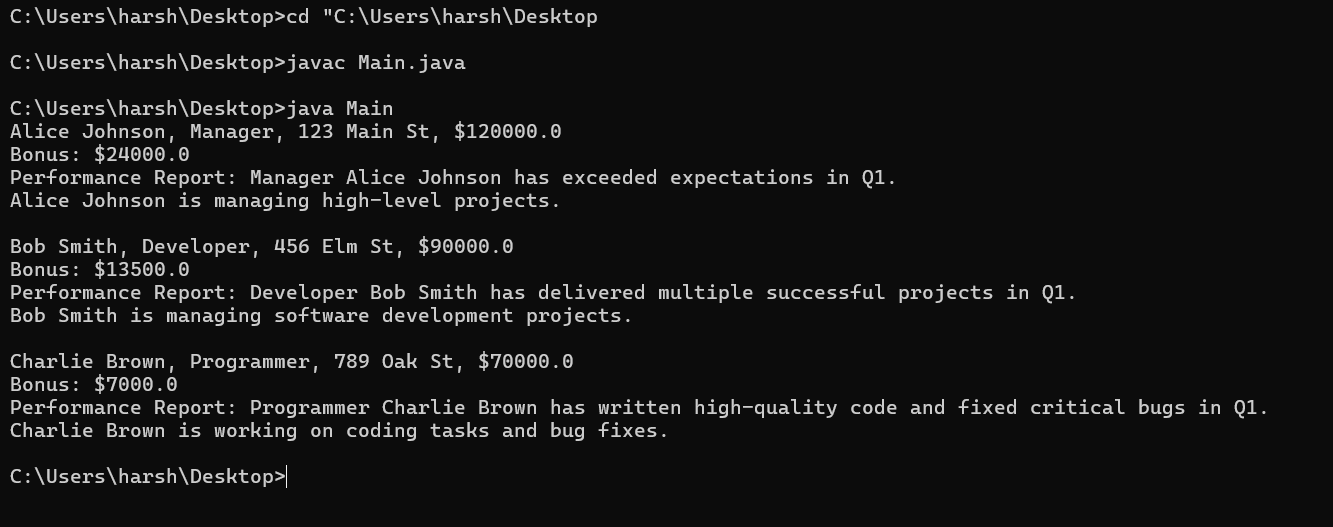
**System.out.println("Performance Report: " + programmer.generatePerformanceReport());**

**programmer.manageProjects();**

**}**

**}**

**Output:**

****