

**NAME:**

HAMZA IMRAN

**ROLL NO.:**

SP25-BCS-043

**SUBJECT:**

Object Oriented Programming

**TOPIC:**

ASSIGNMENT # 02

**SUBMITTED TO:**

SHAHID BHATTI

**SUBMISSION DATE:**

OCTOBER 29,2025

**CLASSES**

# ENUMS

public enum PlotType{

    RES\_5\_MARLA(4000000),

    RES\_10\_MARLA(7500000),

    RES\_1\_KANAL(14000000),

    COMM\_SHOP(3000000),

    COMM\_OFFICE(5000000),

    PARKING(200000);

    private final double Price;

    PlotType(int Price){

    this.Price=Price;

}

    public double getPrice(){

    return Price;

}

}

public enum PlotType{

    RES\_5\_MARLA(4000000),

    RES\_10\_MARLA(7500000),

    RES\_1\_KANAL(14000000),

    COMM\_SHOP(3000000),

    COMM\_OFFICE(5000000),

    PARKING(200000);

    private final double Price;

    PlotType(int Price){

    this.Price=Price;

}

    public double getPrice(){

    return Price;

}

}

# Plot

public class Plot {

        private String plotID;

        private PlotType type;

        private PlotShape shape;

        private double width;

        private double depth;

        private double front;

        private double back;

        private double area;

        private double price;

        private boolean isAvailable;

        public Plot(String id, PlotType type, PlotShape shape, double width, double depth, double front, double back)

        {

            this.plotID = id;

            this.type = type;

            this.shape = shape;

            this.width = width;

            this.depth = depth;

            this.front = front;

            this.back = back;

            this.price=type.getPrice();

            this.isAvailable=true;

            calculateArea();

        }

        public double getPrice() {

        return price;

        }

        public void calculateArea(){

            area=depth\*width;

        }

        public String getPlotID(){

            return plotID;

        }

        public double getArea(){

            return area;

        }

        public boolean isAvailable(){

            return isAvailable;

        }

        public boolean BookPlot(){

            if(isAvailable==true){

                isAvailable=false;

                System.out.println("Plot booked successfully");

                return true;

            }

            else{

                System.out.println("Plot is already booked");

                return false;

            }

        }

        public boolean CancelBooking(){

            if(isAvailable==false){

                isAvailable=true;

                System.out.println("Plot cnacelled successfully");

                return true;

            }

            else{

                System.out.println("Plot is already available");

                return false;

            }

        }

        @Override

public String toString() {

    String availability;

    if (isAvailable == true) {

        availability = "Available";

    } else {

        availability = "Booked";

    }

    return String.format("%-10s %-20s %-15.2f %-20.2f  %-12s",

            plotID,

            type,

            price,

            area,

            availability); }

}

# Corner Plot

public class Cornerplot extends Plot{

    private double cornercharge;

    public Cornerplot(String id, PlotType type, PlotShape shape, double width, double depth, double front, double back) {

        super(id, type, shape, width, depth, front, back);

        CalculateExtraCharge();;

    }

    public void CalculateExtraCharge(){

        cornercharge=(getPrice()\*8)/100;

    }

     public double getCornerCharge() {

        return cornercharge;

    }

    public boolean BookPlot(){

        if(isAvailable()==true){

              double totalPrice = getPrice() + cornercharge;

        System.out.println("Price of CornerPlot: " + totalPrice);

        return super.BookPlot();

    }

    else {

        System.out.println("Corner plot is already booked");

        return false;

    }

    }

       public boolean CancelBooking() {

        if (isAvailable() == false) {

            System.out.println("Corner plot booking cancelled successfully");

            return super.CancelBooking();

        }

        else {

            System.out.println("Corner plot is already available");

            return false;

        }

    }

 }

# SHOP CLASS

public class Shop extends commercialPlot{

    private String shopNumber;

    private double shopCharge;

    public Shop(String id, PlotType type, PlotShape shape, double width, double depth, double front, double back, String shopNumber) {

        super(id, type, shape, width, depth, front, back);

        this.shopNumber = shopNumber;

        calculateShopCharge();

    }

    public void calculateShopCharge() {

        shopCharge = super.getPrice() \* 0.05;

    }

    @Override

    public boolean BookPlot() {

        if (isAvailable() == true) {

            double totalPrice = super.getPrice() + (super.getPrice() \* 0.10) + shopCharge;

            System.out.println("Shop " + shopNumber + " booked successfully ");

            System.out.println("Total Price: " + totalPrice);

            return super.BookPlot();

        }

        else {

            System.out.println("Shop " + shopNumber + " is already booked.");

            return false;

        }

    }

  @Override

    public String toString() {

        String info = "";

        info += super.toString() + "\n";

        info += "Shop Number: " + shopNumber + "\n";

        info += "Shop Extra Charge (5%): " + shopCharge;

        return info;

    }

}

# COMMERCIAL MARKET

public class CommercialMarket {

    private String marketName;

    private Shop[] shops;

    private int shopCount;

    public CommercialMarket(String name, int totalShops) {

        this.marketName = name;

        this.shops = new Shop[totalShops];

        this.shopCount = 0;

    }

    public void addShop(Shop s) {

        if (shopCount < shops.length) {

            shops[shopCount] = s;

            shopCount++;

        } else {

            System.out.println("Market is full, cannot add more shops.");

        }

    }

    public void displayMarketInfo() {

        System.out.println("Market Name: " + marketName);

        System.out.println("Total Shops: " + shopCount);

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

            System.out.println(shops[i].toString());

        }

    }

    public Shop findFirstAvailableShop() {

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

            if (shops[i].isAvailable() == true) {

                return shops[i];

            }

        }

        return null;

    }

    public boolean bookShop(String id) {

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

            if (shops[i].getPlotID().equals(id)) {

                return shops[i].BookPlot();

            }

        }

        System.out.println("Shop with ID " + id + " not found.");

        return false;

    }

    public boolean cancelBooking(String id) {

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

            if (shops[i].getPlotID().equals(id)) {

                return shops[i].CancelBooking();

            }

        }

        System.out.println("Shop with ID " + id + " not found.");

        return false;

    }

}

# PARK CLASS

public class Park {

    private String parkName;

    private PlotShape shape;

    private double width;

    private double depth;

    private double area;

    public Park(String parkName, PlotShape shape, double width, double depth) {

        this.parkName = parkName;

        this.shape = shape;

        this.width = width;

        this.depth = depth;

        calculateArea();

    }

    public void calculateArea() {

        if (shape == PlotShape.RECTANGLE) {

            area = width \* depth;

        }

        else if (shape == PlotShape.TRAPEZOID) {

            area = ((width + depth) / 2) \* 10;

        }

        else {

            area = (width \* 5) + (depth \* 5);

        }

    }

    public String getParkName() {

        return parkName;

    }

    public double getArea() {

        return area;

    }

    public void displayInfo() {

        System.out.println("Park Name: " + parkName);

        System.out.println("Shape: " + shape);

        System.out.println("Width: " + width + ", Depth: " + depth);

        System.out.println("Area: " + area + " sq units");

    }

}

# BLOCK CLASS

public class Block {

    private String blockName;

    private Plot[] plots;

    private int plotCount;

    private CommercialMarket market;

    private ParkingPlot parking;

    public Block(String blockName, int totalPlots) {

        this.blockName = blockName;

        this.plots = new Plot[totalPlots];

        this.plotCount = 0;

    }

    public void addPlot(Plot p) {

        if (plotCount < plots.length) {

            plots[plotCount] = p;

            plotCount++;

        } else {

            System.out.println("No more space to add new plots in " + blockName);

        }

    }

    public void setCommercialMarket(CommercialMarket market) {

        this.market = market;

    }

    public void setParkingPlot(ParkingPlot parking) {

        this.parking = parking;

    }

    public void displayBlockInfo() {

    System.out.println("Block Name: " + blockName);

    System.out.println("Total Plots: " + plotCount);

    System.out.printf("%-10s %-20s %-15s %-15s %-15s\n",

            "PLOT ID", "PLOT TYPE", "PLOT PRICE", "PLOT AREA", "AVAILABILITY");

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

        System.out.println(plots[i].toString());

    }

}

    public Plot findAvailablePlot() {

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

            if (plots[i].isAvailable() == true) {

                return plots[i];

            }

        }

        return null;

    }

    public boolean bookPlot(String id) {

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

            if (plots[i].getPlotID().equals(id)) {

                return plots[i].BookPlot();

            }

        }

        System.out.println("Plot with ID " + id + " not found.");

        return false;

    }

    public boolean cancelBooking(String id) {

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

            if (plots[i].getPlotID().equals(id)) {

                return plots[i].CancelBooking();

            }

        }

        System.out.println("Plot with ID " + id + " not found.");

        return false;

    }

}

# HOUSING SOCIETY

public class Society {

    private String societyName;

    private Plot[] plots;

    private int plotCount;

    public Society(String societyName,int size) {

        this.societyName = societyName;

        plots = new Plot[size];

        plotCount = 0;

    }

    public String getSocietyName() {

        return societyName;

    }

    public void setSocietyName(String societyName) {

        this.societyName = societyName;

    }

    public Plot[] getPlots() {

        return plots;

    }

    public int getPlotCount() {

        return plotCount;

    }

    public void addPlot(Plot plot) {

        if (plotCount < plots.length) {

            plots[plotCount] = plot;

            plotCount++;

            System.out.println("Plot added successfully in " + societyName);

        } else {

            System.out.println("Cannot add more plots. Society is full.");

        }

    }

    public void showAllPlots() {

    if (plotCount == 0) {

        System.out.println("No plots available in " + societyName);

    } else {

        System.out.println("Plots in " + societyName + ":");

        System.out.printf("%-10s %-20s %-15s %-15s %-15s\n",

                "PLOT ID", "PLOT-TYPE", "PLOT-PRICE", "PLOT-AREA", "AVAILABILITY");

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

            System.out.println(plots[i].toString());

        }

    }

}

    public void bookPlotByID(String id) {

        boolean found = false;

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

            if (plots[i].getPlotID().equals(id)) {

                found = true;

                plots[i].BookPlot();

                break;

            }

        }

        if (found == false) {

            System.out.println("Plot ID not found in " + societyName);

        }

    }

    public void cancelPlotByID(String id) {

        boolean found = false;

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

            if (plots[i].getPlotID().equals(id)) {

                found = true;

                plots[i].CancelBooking();

                break;

            }

        }

        if (found == false) {

            System.out.println("Plot ID not found in " + societyName);

        }

    }

    @Override

public String toString() {

    StringBuilder sb = new StringBuilder();

    sb.append("Street: 4\n");

    sb.append(String.format("%-10s %-20s %-15s %-15s %-15s\n",

            "PLOT ID", "PLOT-TYPE", "PLOT-PRICE", "PLOT-AREA", "AVAILABILITY"));

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

        sb.append(plots[i].toString()).append("\n");

    }

    return sb.toString();

}

}

# CITY HOUSING

public class CityHousing {

    private String cityName;

    private Society[] societies;

    private int societyCount;

    public CityHousing(String cityName, int size) {

        this.cityName = cityName;

        societies = new Society[size];

        societyCount = 0;

    }

    public String getCityName() {

        return cityName;

    }

    public void setCityName(String cityName) {

        this.cityName = cityName;

    }

    public void addSociety(Society society) {

        if (societyCount < societies.length) {

            societies[societyCount] = society;

            societyCount++;

            System.out.println("Society added successfully in " + cityName);

        } else {

            System.out.println("Cannot add more societies. City is full.");

        }

    }

    public void bookPlot(String societyName, String plotId) {

        boolean found = false;

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

            if (societies[i].getSocietyName().equals(societyName)) {

                found = true;

                societies[i].bookPlotByID(plotId);

                break;

            }

        }

        if (found == false) {

            System.out.println("Society not found in " + cityName);

        }

    }

    public void cancelPlot(String societyName, String plotId) {

        boolean found = false;

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

            if (societies[i].getSocietyName().equals(societyName)) {

                found = true;

                societies[i].cancelPlotByID(plotId);

                break;

            }

        }

        if (found == false) {

            System.out.println("Society not found in " + cityName);

        }

    }

    public void showAllSocieties() {

        if (societyCount == 0) {

            System.out.println("No societies available in " + cityName);

        } else {

            System.out.println("\nCity: " + cityName);

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

                societies[i].showAllPlots();

            }

        }

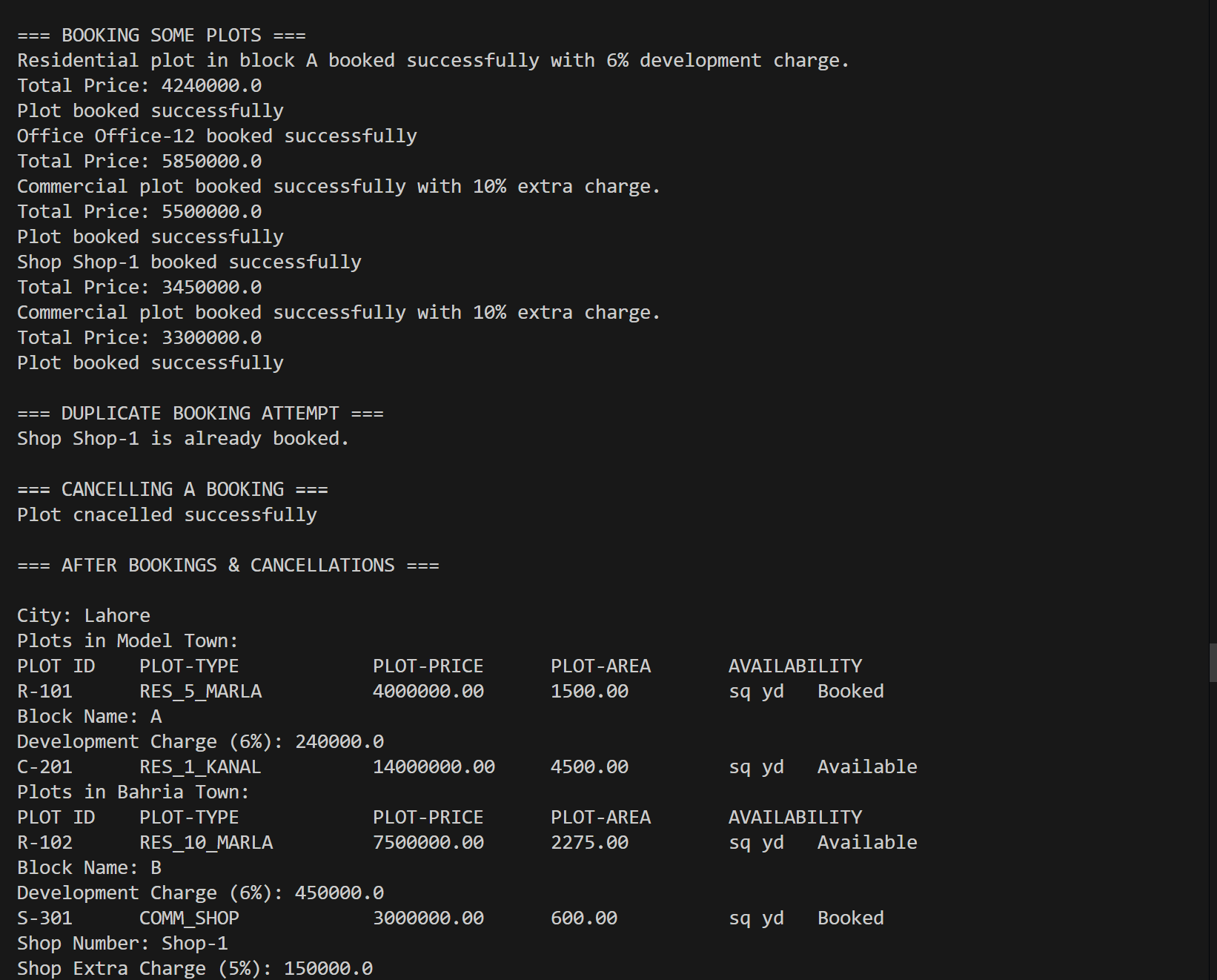
    }

}

# OUTPUT

**A screenshot of a computer screen

AI-generated content may be incorrect.**

****

A screenshot of a computer screen

AI-generated content may be incorrect.

# 