**Question One. Total Marks     /10 (2.5 Marks each)**

Suppose you are designing a program for a juice shop that sells different types of juices. You should create a parent juice class with common members, and three child classes for the three main juice categories: FruitJuice, VegetableJuice, and MixedJuice. Each of these child classes should have their own sub-classes, such as CitrusJuice, BerryJuice, and TropicalJuice for FruitJuice, and LeafyGreenJuice, RootVegetableJuice, and MixedVegetableJuice for VegetableJuice. Finally, the MixedJuice class can have its own sub-classes such as FruitAndVeggieJuice and SmoothieJuice. Fruit juices should have additional variable of *season*. Vegetable juices have *origin* string member.

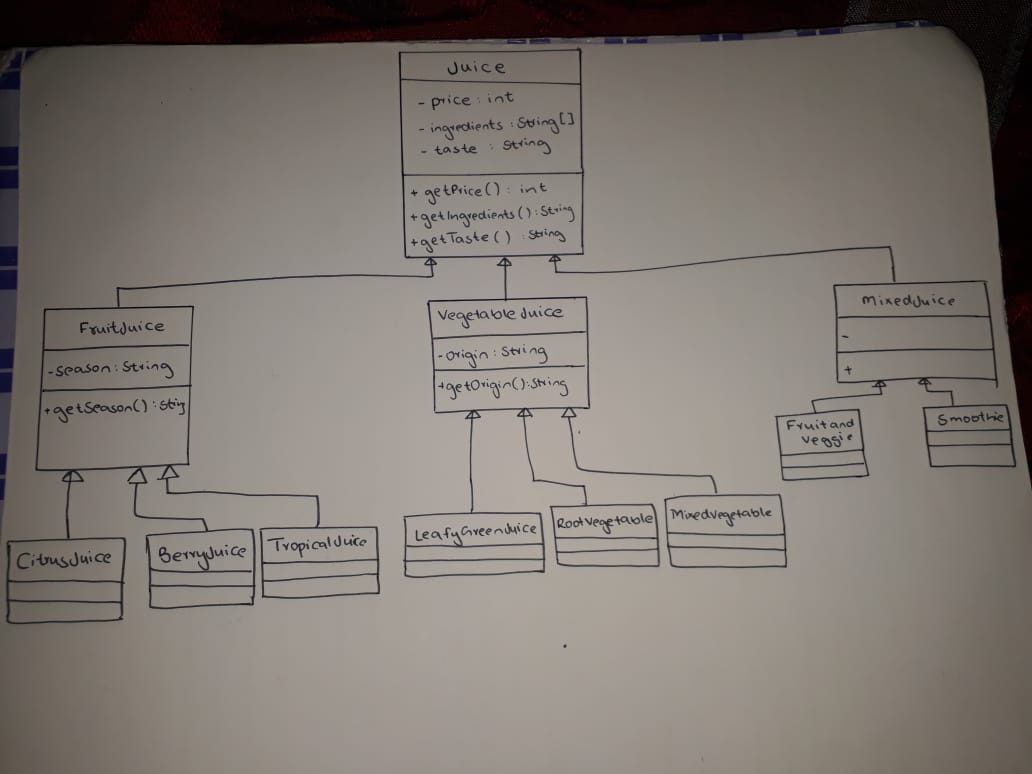
All the juice classes have common members such as price, ingredients, and taste. For example, a citrus fruit juice might have oranges, lemons, and grapefruits as its ingredients, a price of 399, and a sweet and tangy taste. Similarly, a leafy green vegetable juice might have kale, spinach, and celery as its ingredients, a price of 499, and a bitter and earthy taste.

On the other hand, juice shops are also keeping track of Sales. Each sale has members of sale ID, CustomerName, total bill, numberOfFuitJuice, numberOfVegetableJuice and numberOfMixedJuice.

1. Draw a class diagram to represent the defined hierarchy. Write the code to design these classes with proper inheritance. Use constructors to assign initial values and have setters/getters to change at a later stage. The program should be able to track the number of objects created in each class.
2. Include a print function in parent class that prints the details of the objects. Override the print function in all child classes where the function first call parent print function and subsequently print additional information. For example, object of citrus fruit juice call parent function to print its values but it also print specialized values of season as well. Each function should print a statement with your name and students id as such: “22k-1234 Fahim also likes citrus fruit juice”.
3. The main function should welcome the customer and get name first. Afterwards, it presents menu to the user where first menu has major categories and after selection, the menu presents the sub categories. After selection, an object of the selected juice is created. The process repeats until the user enter selects check out from the menu.
4. Write a global overloaded sum() function, the function should be able to add two sales objects and return updated sale object with sum of both bills, and juices count. The function should make sure that the customer name in both the sales is the same. The following function should work SALES ob1, ob2; SALES ob3 = ob1.sum(ob2);

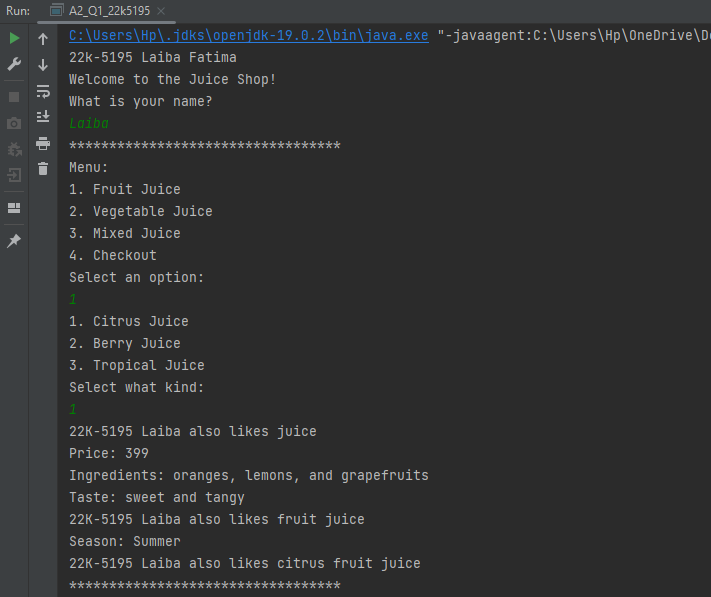
*Solution:*

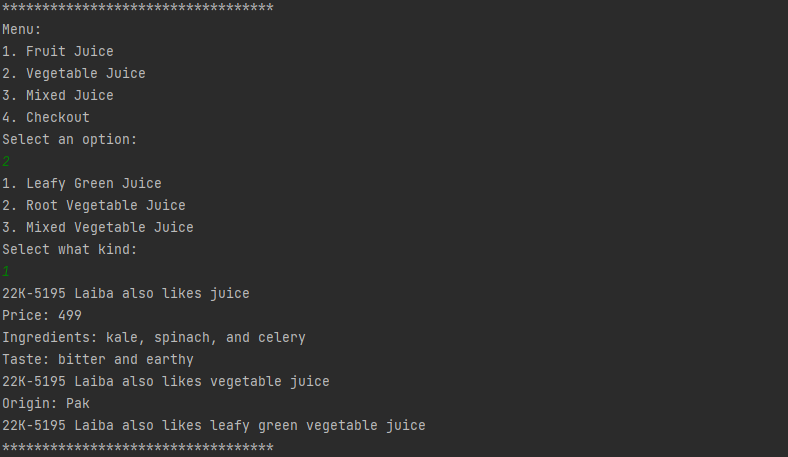
a) Class Diagram

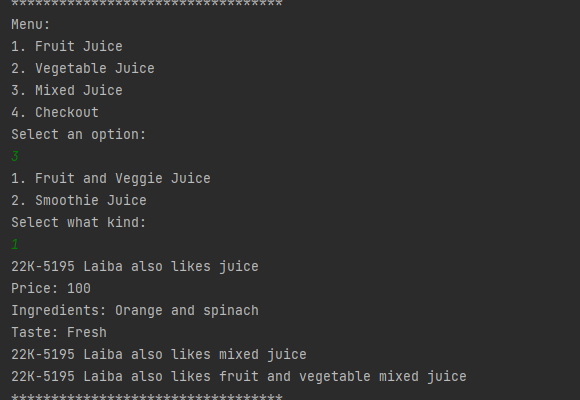


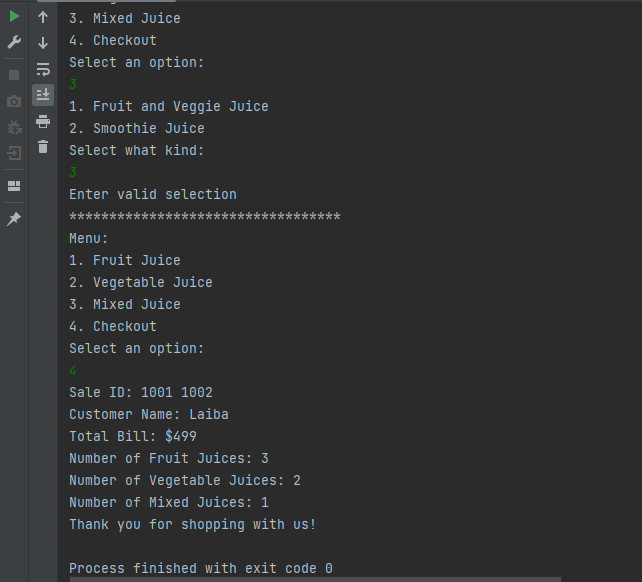
//22k-5195 Laiba Fatima  
import java.util.Scanner;  
  
//Class diagram is in the Word document  
class Juice{  
 private int price;  
 private String ingredients;  
 private String taste;  
 public static int *countjuice* = 0; //tracking number of objects created  
  
 public Juice(){}  
 public Juice(int price, String ingredients, String taste) {  
 this.price = price;  
 this.ingredients = ingredients;  
 this.taste = taste;  
 *countjuice*++;  
 }  
  
 public static int getCountjuice() {  
 return *countjuice*;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 public String getIngredients() {  
 return ingredients;  
 }  
  
 public void setIngredients(String ingredients) {  
 this.ingredients = ingredients;  
 }  
  
 public String getTaste() {  
 return taste;  
 }  
  
 public void setTaste(String taste) {  
 this.taste = taste;  
 }  
  
 public void print(){  
 System.*out*.println("22K-5195 Laiba also likes juice");  
 System.*out*.println("Price: " + price);  
 System.*out*.println("Ingredients: "+ingredients);  
 System.*out*.println("Taste: " +taste);  
 }  
  
}  
class FruitJuice extends Juice {  
 private String season;  
 public static int *countfruit*= 0;  
  
 public FruitJuice(int price, String ingredients, String taste, String season) {  
 super(price, ingredients, taste);  
 this.season = season;  
 *countfruit*++;  
 }  
  
 public static int getCountfruit() {  
 return *countfruit*;  
 }  
  
 public String getSeason() {  
 return season;  
 }  
  
 public void setSeason(String season) {  
 this.season = season;  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes fruit juice");  
 System.*out*.println("Season: " +season);  
 }  
}  
class VegetableJuice extends Juice{  
 private String origin;  
  
 public VegetableJuice(int price, String ingredient, String taste, String origin) {  
 super(price, ingredient, taste);  
 this.origin = origin;  
  
 }  
  
 public String getOrigin() {  
 return origin;  
 }  
  
 public void setOrigin(String origin) {  
 this.origin = origin;  
 }  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes vegetable juice");  
 System.*out*.println("Origin: " +origin);  
 }  
}  
class MixedJuice extends Juice{  
  
  
 public MixedJuice(int price, String ingredient, String taste) {  
 super(price, ingredient, taste);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes mixed juice");  
 }  
}  
class CitrusJuice extends FruitJuice{  
  
 public CitrusJuice(int price, String ingredients, String taste, String season) {  
 super(price, ingredients, taste, season);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes citrus fruit juice");  
 }  
}  
class BerryJuice extends FruitJuice{  
  
 public BerryJuice(int price, String ingredients, String taste, String season) {  
 super(price, ingredients, taste, season);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes berry fruit juice");  
 }  
}  
class TropicalJuice extends FruitJuice{  
 public TropicalJuice(int price, String ingredients, String taste, String season) {  
 super(price, ingredients, taste, season);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes tropical juice");  
 }  
}  
class LeafyGreenJuice extends VegetableJuice{  
 public LeafyGreenJuice(int price, String ingredient, String taste, String origin) {  
 super(price, ingredient, taste, origin);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes leafy green vegetable juice");  
 }  
}  
class RootVegetableJuice extends VegetableJuice{  
  
 public RootVegetableJuice(int price, String ingredient, String taste, String origin) {  
 super(price, ingredient, taste, origin);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes root vegetable juice");  
 }  
}  
class MixedVegetableJuice extends VegetableJuice{  
  
 public MixedVegetableJuice(int price, String ingredient, String taste, String origin) {  
 super(price, ingredient, taste, origin);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes mixed vegetable juice");  
 }  
}  
class FruitAndVeggieJuice extends MixedJuice{  
  
 public FruitAndVeggieJuice(int price, String ingredient, String taste) {  
 super(price, ingredient, taste);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes fruit and vegetable mixed juice");  
 }  
}  
class SmoothieJuice extends MixedJuice{  
  
 public SmoothieJuice(int price, String ingredient, String taste) {  
 super(price, ingredient, taste);  
 }  
  
 public void print() {  
 super.print();  
 System.*out*.println("22K-5195 Laiba also likes smoothie mixed juice");  
 }  
}  
class Sale extends Juice{  
 private String saleID;  
 private String CustomerName;  
 public static int *totalbill*;  
 private int numberOfFruitJuice;  
 private int numberOfVegetableJuice;  
 private int numberOfMixedJuice;  
  
  
 public Sale(String id, String name, int bill, int fruitno, int vegno, int mixno) {  
 this.saleID = id;  
 this.CustomerName = name;  
 this.*totalbill* = bill;  
 this.numberOfFruitJuice = fruitno;  
 this.numberOfVegetableJuice = vegno;  
 this.numberOfMixedJuice = mixno;  
 }  
  
 public static float totalbill() {  
  
 return *totalbill*;  
 }  
  
 public String getSaleID() {  
 return saleID;  
 }  
  
 public void setSaleID(String id) {  
 this.saleID = id;  
 }  
  
 public String getCustomerName() {  
 return CustomerName;  
 }  
  
 public void setCustomerName(String name) {  
 CustomerName = name;  
 }  
  
 public int getTotalbill() {  
 return *totalbill*;  
 }  
  
 public void setTotalbill(int bill) {  
 this.*totalbill* = bill;  
 }  
  
 public int getNumberOfFruitJuice() {  
 return numberOfFruitJuice;  
 }  
  
 public void setNumberOfFruitJuice(int fruitno) {  
 this.numberOfFruitJuice = fruitno;  
 }  
  
 public int getNumberOfVegetableJuice() {  
 return numberOfVegetableJuice;  
 }  
  
 public void setNumberOfVegetableJuice(int vegno) {  
 this.numberOfVegetableJuice = vegno;  
 }  
  
 public int getNumberOfMixedJuice() {  
 return numberOfMixedJuice;  
 }  
  
 public void setNumberOfMixedJuice(int mixno) {  
 this.numberOfMixedJuice = mixno;  
 }  
  
 public void print() {  
 System.*out*.println("Sale ID: " + saleID);  
 System.*out*.println("Customer Name: " + CustomerName);  
 System.*out*.println("Total Bill: $" + *totalbill*);  
 System.*out*.println("Number of Fruit Juices: " + numberOfFruitJuice);  
 System.*out*.println("Number of Vegetable Juices: " + numberOfVegetableJuice);  
 System.*out*.println("Number of Mixed Juices: " + numberOfMixedJuice);  
 }  
  
 public static Sale sum(Sale ob1, Sale ob2) {  
 if (ob1.getCustomerName().equals(ob2.getCustomerName())) {  
 String saleID = ob1.getSaleID() + " " + ob2.getSaleID();  
 String customerName = ob1.getCustomerName();  
 int totalBill = ob1.getTotalbill() + ob2.getTotalbill();  
 int numberOfFruitJuice = ob1.getNumberOfFruitJuice() + ob2.getNumberOfFruitJuice();  
 int numberOfVegetableJuice = ob1.getNumberOfVegetableJuice() + ob2.getNumberOfVegetableJuice();  
 int numberOfMixedJuice = ob1.getNumberOfMixedJuice() + ob2.getNumberOfMixedJuice();  
 Sale ob3 = new Sale(saleID, customerName, *totalbill*, numberOfFruitJuice, numberOfVegetableJuice, numberOfMixedJuice);  
 return ob3;  
 } else {  
 System.*out*.println("Error: Different customer names.");  
 return null;  
 }  
  
 }  
}  
public class A2\_Q1\_22k5195 {  
 public static void main(String[] args) {  
 Scanner a = new Scanner(System.*in*);  
 System.*out*.println("22k-5195 Laiba Fatima");  
 System.*out*.println("Welcome to the Juice Shop!");  
 System.*out*.println("What is your name?");  
  
 String name = a.next();  
  
 do {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println("Menu: ");  
 System.*out*.println("1. Fruit Juice");  
 System.*out*.println("2. Vegetable Juice");  
 System.*out*.println("3. Mixed Juice");  
 System.*out*.println("4. Checkout");  
 System.*out*.println("Select an option: ");  
 int choice = a.nextInt();  
 switch (choice) {  
 case 1:  
 System.*out*.println("1. Citrus Juice");  
 System.*out*.println("2. Berry Juice");  
 System.*out*.println("3. Tropical Juice");  
 System.*out*.println("Select what kind:");  
 int select = a.nextInt();  
 switch (select){  
 case 1:  
 CitrusJuice c = new CitrusJuice(399,"oranges, lemons, and grapefruits","sweet and tangy", "Summer");  
 c.print();  
 break;  
 case 2:  
 BerryJuice b = new BerryJuice(100,"strawberry","sour", "Winter");  
 b.print();  
 break;  
 case 3:  
 TropicalJuice t = new TropicalJuice(200,"Coconut", "Sweet", "Summer");  
 t.print();  
 break;  
 default:  
 System.*out*.println("Enter valid selection");  
 }  
 break;  
 case 2:  
 System.*out*.println("1. Leafy Green Juice");  
 System.*out*.println("2. Root Vegetable Juice");  
 System.*out*.println("3. Mixed Vegetable Juice");  
 System.*out*.println("Select what kind:");  
 select = a.nextInt();  
 switch (select){  
 case 1:  
 LeafyGreenJuice l = new LeafyGreenJuice(499,"kale, spinach, and celery", "bitter and earthy", "Pak");  
 l.print();  
 break;  
 case 2:  
 RootVegetableJuice r = new RootVegetableJuice(100,"root beer","bitter", "India");  
 r.print();  
 break;  
 case 3:  
 MixedVegetableJuice m = new MixedVegetableJuice(200,"all the veggies", "bitter", "UK");  
 m.print();  
 break;  
 default:  
 System.*out*.println("Enter valid selection");  
 }  
 break;  
 case 3:  
 System.*out*.println("1. Fruit and Veggie Juice");  
 System.*out*.println("2. Smoothie Juice");  
 System.*out*.println("Select what kind:");  
 select = a.nextInt();  
 switch (select){  
 case 1:  
 FruitAndVeggieJuice f = new FruitAndVeggieJuice(100, "Orange and spinach", "Fresh");  
 f.print();  
 break;  
 case 2:  
 SmoothieJuice s = new SmoothieJuice(100,"strawberry, lemon, apple","Sweet");  
 s.print();  
 break;  
 default:  
 System.*out*.println("Enter valid selection");  
 }  
 break;  
 case 4:  
 float bill;  
 bill= Sale.*totalbill*();  
 System.*out*.println("Total bill: "+bill);  
  
 System.*out*.println("Thank you for shopping with us! " );  
 //implementation of d part  
 Sale sale1 = new Sale("1001", "Laiba", 399, 1, 1, 1);  
 Sale sale2 = new Sale("1002", name, 499, 2, 1, 0);  
  
 // Sum two sales objects  
 Sale sumSale = Sale.*sum*(sale1, sale2);  
 if (sumSale != null) {  
 sumSale.print();  
 }  
 System.*exit*(0);  
 break;  
 default:  
 System.*out*.println("Enter valid choice");  
 }  
 } while (true);  
   
 }  
}

Output:









**Question Two. Total Marks     /10**

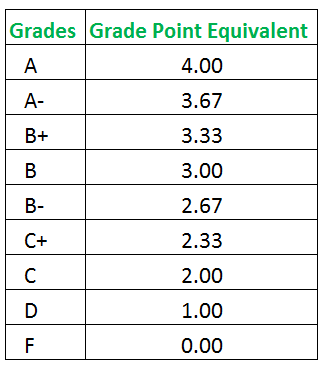
Your University is updating software for grading and student record saving. The grading report is prepared by the administration based on the payment of tuition fees by individual student. Grades are awarded and shown on report if student semester fees is paid. If in case of not paid student is grades are removed from report as well as late penalty charges along with fees are charged.

**Courses:                                                                                                                             (1 Mark)**

Attributes of courses will be course name, course code, credit hours. The functionality for courses will be:

1. Set the course information. (Contain all attributes of course)
2. Print the course information.
3. Show the credit hours.
4. Show the course number

**Student:                                                                                                         7 Marks (0.5+1+0.5+2+3)**

Attributes of student will be ID, student name, number of courses enrolled, course name, grade for each course. A variable indicating whether the student had paid fees or not. Every student is a person, and every student takes courses**.** The basic operations to be performed on an object of type student are as follows: 

1. Set the student information.
2. Print the student information.
3. Calculate the number of credit hours taken.
4. Calculate the GPA. Take marks of N **courses**,
5. Calculate the GPA and the GPA percentage of the student.
6. Calculate the semester fees**.**

**In case of nonpayment 5% of per credit fees is additionally included in fees voucher.**

Check whether the student is enrolled in particular course of not. Create a function that prints the result grades. If the student has paid semester fees, the grades and the GPA are shown. Otherwise, **“LOCKED GRADES”** are printed in place of each grade, the GPA is not shown, a message indicates that the grades are being held for nonpayment of the Semester fees, due amount along with penalty charges are shown.       **(2 Marks)**

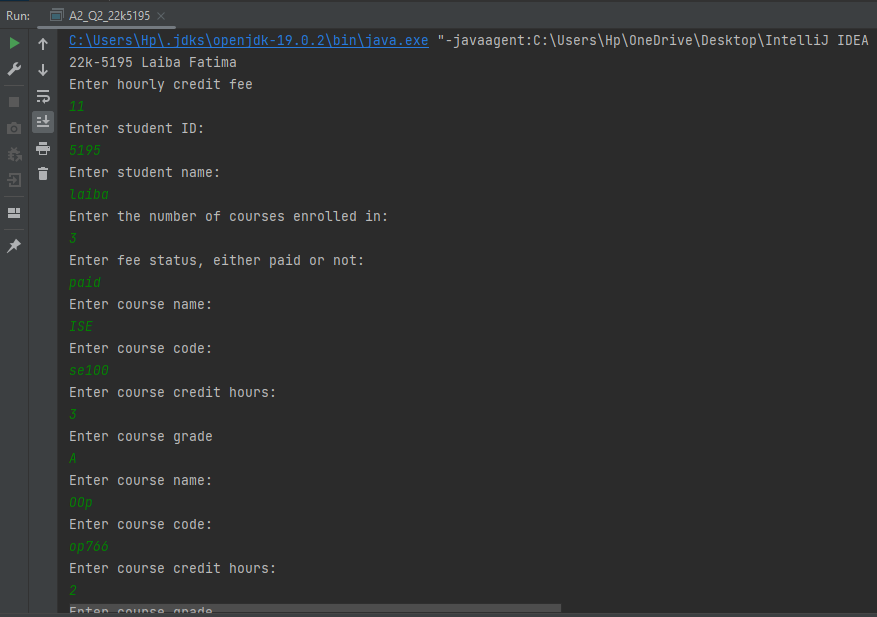
For test program create main function take your roll number as input for example: K22-0328. The batch code divided by half gives per credit hour charge for the students of year. Here 22k will give 11 as result. Later create **n** array of students. The **n** is second digit of your roll number. The first line indicates the number of students enrolled and the tuition rate per credit hour. The students’ data is given thereafter.

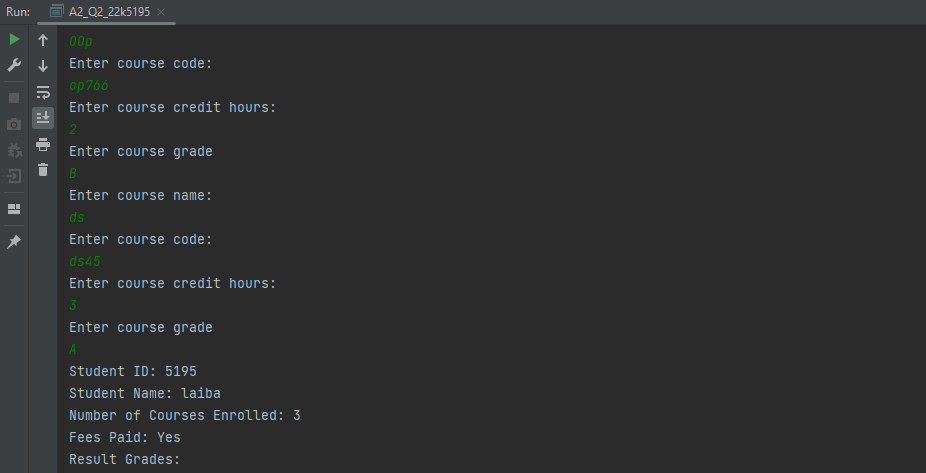
|  |  |
| --- | --- |
| **Sample input:**  3 11000  AMMAR K22-0933       Y 4  DLD EL1004                4 A  OOP CS2004                3 B  Communication Skills CS431    3 B  Islamiyat IS356             3 A  AMMAR ID is K22-0328, student has paid the tuition indicated in first line, and he is taking four courses.  **OUTPUT SAMPLE 1**  Student Name: AMMAR  Student ID: K22-0933   Number of courses enrolled: 4  Course No Course Name Credits Grade  DLD  EL1004                4 A  OOP CS2004                3 B  Communication Skills CS431    3 B  Islamiyat  IS356             3 A  Total number of credit hours: 13  Semester GPA: 3.54 | **OUTPUT SAMPLE 2**  Student Name: Bilal  Student ID: 20K-0978  Number of courses enrolled: 5  Course No Course Name Credits Grade  Calculus CaO234 4 **LOCKED GRADES**  DlD EL1004    4 **LOCKED GRADE**  English ENG378 3 **LOCKED GRADE**  Mathematics MTH346 3 **LOCKED GRADE**  Data Structures CS2008 3 **LOCKED GRADE**  Total number of credit hours: 17  \*\*\* Grades are being held for not paying the tuition. \*\*\*  Amount Due: 5865.00 |

*Solution:*

//22k-5195 Laiba Fatima  
import java.util.Scanner;  
import java.util.ArrayList;  
  
class Courses{  
 public String coursename;  
 public String coursecode;  
 public int credithours;  
  
 public Courses(String name, String code, int hours){  
 this.coursename=name;  
 this. coursecode=code;  
 this.credithours=hours;  
 }  
 public void printCourseInfo() {  
 System.*out*.println("Displaying: ");  
 System.*out*.println("Course Name: "+coursename);  
 System.*out*.println("Course Code: "+coursecode);  
 System.*out*.println("Credit Hours: "+credithours);  
 }  
  
 public void showcredithours() {  
 System.*out*.println("Credit Hours: " + credithours);  
 }  
 public void showcoursecode() {  
 System.*out*.println("Course Code: " + coursecode);  
 }  
}  
class Student {  
 public String studentID;  
 public String studentname;  
 public int enrolledcourses;  
 public String coursename[];  
 public String coursegrade[];  
 public boolean feesPaid;  
  
 public Student(String id, String name, int num, boolean status){  
 studentID=id;  
 studentname=name;  
 enrolledcourses=num;  
 feesPaid=status;  
 }  
 ArrayList<Courses> courseArrayList;  
  
 public ArrayList<Courses> getCourseArrayList() {  
 return courseArrayList;  
 }  
  
 public void setCourseArrayList(ArrayList<Courses> courseArrayList) {  
 this.courseArrayList = courseArrayList;  
 }  
  
 public String[] getCoursegrade() {  
 return coursegrade;  
 }  
  
 public void setCoursegrade(String[] coursegrade) {  
 this.coursegrade = coursegrade;  
 }  
 public int credithourstaken(){  
 int totalcredithours, i, j, ch;  
 totalcredithours =0;  
 j=courseArrayList.size();  
 for(i=0;i<j;i++){  
 ch =courseArrayList.get(i).credithours;  
 totalcredithours = totalcredithours + ch;  
 }  
 return(totalcredithours);  
 }  
  
  
 public void printstudentinfo() {  
 System.*out*.println("Student ID: " + studentID);  
 System.*out*.println("Student Name: " + studentname);  
 System.*out*.println("Number of Courses Enrolled: " + enrolledcourses);  
 System.*out*.println("Fees Paid: " + (feesPaid ? "Yes" : "No"));  
 }  
  
  
  
 public double calculateGPA() {  
 int totalCreditHours = credithourstaken();  
 double gpa = 0.0;  
 for (int i = 0; i < enrolledcourses; i++) {  
 String grade = coursegrade[i];  
 double gradePoint = 0.0;  
  
 if (grade.equals("A")) {  
 gradePoint = 4.00;  
 } else if (grade.equals("A-")) {  
 gradePoint = 3.67;  
 } else if (grade.equals("B+")) {  
 gradePoint = 3.33;  
 } else if (grade.equals("B")) {  
 gradePoint = 3.00;  
 } else if (grade.equals("B-")) {  
 gradePoint = 2.67;  
 } else if (grade.equals("C+")) {  
 gradePoint = 2.33;  
 } else if (grade.equals("C")) {  
 gradePoint = 2.00;  
 } else if (grade.equals("D")) {  
 gradePoint = 1.00;  
 } else if (grade.equals("F")) {  
 gradePoint = 0.0;  
 }  
 gpa += gradePoint;  
 }  
 gpa /= enrolledcourses;  
 return gpa;  
 }  
  
 public double calculateGPAPercentage() {  
 double gpa = calculateGPA();  
 return gpa \* 9.5;  
 }  
  
 public double calculateSemesterFees(double perCreditFee) {  
 double totalFees = 0.0;  
 int totalCreditHours = credithourstaken();  
 double perCreditFees = perCreditFee;  
 totalFees = totalCreditHours \* perCreditFees;  
 if (!feesPaid) {  
 perCreditFees = perCreditFees \* 0.05;  
 totalFees += perCreditFees;  
 }  
  
 return totalFees;  
 }  
  
 public boolean isEnrolledInCourse(String courseCode) {  
 for (int i = 0; i < enrolledcourses; i++) {  
 if (coursename[i].equals(courseCode)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public void printResultGrades(boolean feesPaid, double HourlyCreditFee) {  
 System.*out*.println("Result Grades: ");  
 if (feesPaid) {  
 for (int i = 0; i < enrolledcourses; i++) {  
 System.*out*.println("Course Name: " + coursename[i] + ", Grade: " + coursegrade[i]);  
 }  
 System.*out*.println("GPA: " + calculateGPA());  
 System.*out*.println("GPA Percentage: " + calculateGPAPercentage() + "%");  
 } else {  
 System.*out*.println("LOCKED GRADES");  
 System.*out*.println("GPA: N/A");  
 System.*out*.println("Message: Grades are being held for nonpayment of Semester fees.");  
 double semesterFees = calculateSemesterFees(HourlyCreditFee);  
 System.*out*.println("Due Amount: " + semesterFees);  
 }  
 }  
}  
  
 public class A2\_Q2\_22k5195 {  
 public static void main(String[] args) {  
 Scanner a = new Scanner(System.*in*);  
 System.*out*.println("22k-5195 Laiba Fatima");  
 boolean feepaid;  
 ArrayList<Courses> courseArrayList=new ArrayList<Courses>();  
  
 System.*out*.println("Enter hourly credit fee");  
 double credit = a.nextDouble();  
 System.*out*.println("Enter student ID: ");  
 String id=a.next();  
 System.*out*.println("Enter student name: ");  
 String name=a.next();  
 System.*out*.println("Enter the number of courses enrolled in: ");  
 int coursenum=a.nextInt();  
 System.*out*.println("Enter fee status, either paid or not: ");  
 String fee=a.next();  
 if(fee.equals("paid")){  
 feepaid =true;  
 }  
 else{  
 feepaid =false;  
 }  
  
 Student s1=new Student(id, name, coursenum, feepaid);  
 int i;  
 String grade[]=new String[coursenum];  
 for(i=0;i<coursenum;i++){  
 System.*out*.println("Enter course name: ");  
 String course=a.next();  
 System.*out*.println("Enter course code: ");  
 String code=a.next();  
 System.*out*.println("Enter course credit hours: ");  
 int credithours=a.nextInt();  
 System.*out*.println("Enter course grade");  
 grade[i]=a.next();  
 Courses c=new Courses(course, code, credithours);  
 courseArrayList.add(c);  
 }  
 s1.setCourseArrayList(courseArrayList);  
 s1.setCoursegrade(grade);  
  
 s1.printstudentinfo();  
 s1.printResultGrades(feepaid, credit);  
  
// System.out.println("Enter your roll number: (K\_\_-\_\_\_\_)");  
// String rollnumber = a.next();  
//  
// String batchcode = rollnumber.substring(1, 3);  
// Integer batchvalue = Integer.valueOf(batchcode);  
// int perCreditHourCharge = batchvalue / 2;  
//  
// int noofstudents = Integer.valueOf(rollnumber.substring(5, 6));  
// Student[] students = new Student[noofstudents];  
//  
// System.out.println("Enter tution rate per credit hour");  
// int tuitionrate = a.nextInt();  
// int totalTuitionFee = noofstudents \* tuitionrate \* perCreditHourCharge;  
//  
//  
// for (int i = 0; i < noofstudents; i++) {  
// System.out.println("Enter the data for student " + (i + 1) + ":");  
// System.out.print("ID: ");  
// String id = a.nextLine();  
// System.out.print("Name: ");  
// String name = a.nextLine();  
// System.out.println("No of courses enrolled: ");  
// int courses = a.nextInt();  
// System.out.print("Has semester fees paid (true/false): ");  
// boolean feesPaid = a.nextBoolean();  
//  
// students[i] = new Student(id, name, courses, feesPaid);  
//  
//  
// for (int j = 0; j < courses; j++) {  
// System.out.println("Enter details for Course " + (j + 1) + ":");  
// System.out.print("Course Name: ");  
// String courseName = a.next();  
//  
// System.out.print("Grade: ");  
// String grade = a.next();  
//  
// students[i].setCourseAndGrade(j, courseName, grade);  
// }  
// }  
// for (int i = 0; i < noofstudents; i++) {  
// System.out.println("Result Grades for Student " + (i + 1) + ":");  
// students[i].printResultGrades(perCreditHourCharge, tuitionrate);  
// System.out.println();  
 }  
 }

Output:





**Question Three. Total Marks     /10**

You are hired as a software engineer to automate the Retail store system.

Diagram

Description automatically generated

**Part A)**

In this system, there is a base class `Product` for all the products in the store. This class has two data members: `name` and `price`, and two abstract functions: `getDiscountedPrice()` and `printDetails()`. The `getDiscountedPrice()` function returns the discounted price of the product, and the `printDetails()` function prints the details of the product.       **(2 Marks)**

There are three derived classes: `Book`, `Electronic`, and `Clothing`. These classes inherit from the `Product` class and add some additional data members and functions. The `Book` class has an additional data member `author` and overrides the `printDetails()` function to include the author's name. The `Electronic` class has an additional data member `brand`. The `Clothing` class has an additional data member `size` and overrides the `getDiscountedPrice()` function to provide different discounts for clothing items.        **(2 Marks)**

Create an interface name `Customer` for all the customers in the store. This interface has two data members: `name` and `balance`, and a function `buyProduct()` that allows the customer to buy a product. **The name variable must take your roll number as a value**. The `buyProduct()` function takes an object of a `Product` and uses the `getDiscountedPrice()` function of the product to calculate the discounted price. If the customer has enough balance, the function deducts the discounted price from the balance and prints a message saying that the customer has bought the product. If the customer does not have enough balance, the function prints a message saying that the customer cannot buy the product. Create a class to implement the stated functionality provided by the interface. Hint (You can even use the product class, if possible)         **(1 Mark)**

We have also defined a derived interface `VIPCustomer` that inherits from the `Customer` interface and adds no additional data members but overrides the `buyProduct()` function to provide additional discounts for VIP customers.         **(1 Mark)**

Finally, in the `main () ` function, we have created some products and customers and tested the program by allowing customers to buy products and printing relevant details.                  **(1 Mark)**

**Part B)**

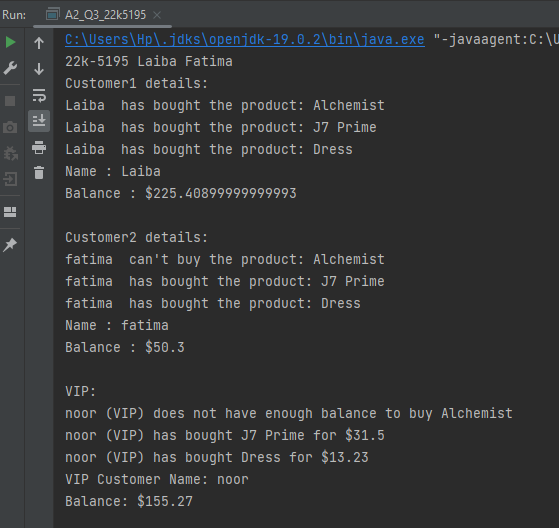
The add function is overloaded as a member function for the Product class. The add() function takes a constant reference to another Product object. It calculates the average price of the two products and creates a new Product object with a concatenated name and the new price. The new product is then returned by the function.                   **(2 Marks)**

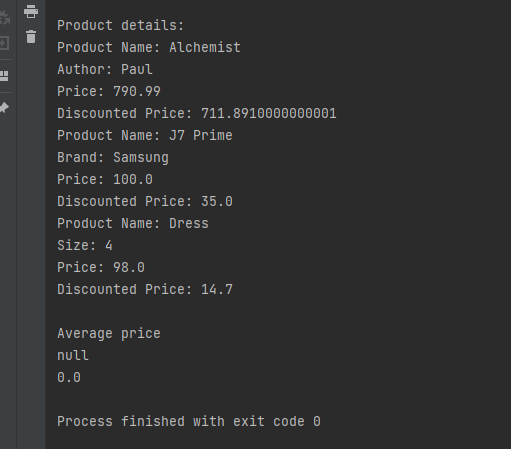
In the main() function, we create two Product objects p1 and p2 with prices of 10 and 2 respectively. We then use the overloaded function to create a new Product object p3, which has a concatenated name of "Book & Pen" and an average price of 6. We finally call printDetails() on p3 to verify that the new product has been created successfully.

*Solution:*

//22k-5195 Laiba Fatima  
abstract class Product {  
 protected String name;  
 protected double price;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public abstract double getDiscountedPrice();  
  
 public abstract void printDetails();  
 public Product() {  
  
 }  
  
 public Product(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
 public Product add(Product p1, Product p2){  
 //applying this function for Book class  
 Product p3=new Book();  
 p3.price=(p1.price+p2.price)/2;  
 p3.name=p1.name+" & "+p2.name;  
 return(p3);  
 }  
}  
class Book extends Product{  
 public String author;  
 public double getDiscountedPrice() {  
 return price \* 0.9 ;  
 }  
  
 public Book() {  
  
 }  
  
 public Book(String name, double price, String author) {  
 super(name, price);  
 this.author = author;  
 }  
  
 @Override  
 public void printDetails() {  
 System.*out*.println("Product Name: " + name);  
 System.*out*.println("Author: " + author);  
 System.*out*.println("Price: " + price);  
 System.*out*.println("Discounted Price: " + getDiscountedPrice());  
 }  
  
}  
class Electronic extends Product{  
 private String brand;  
 public double getDiscountedPrice() {  
  
 return price \* 0.35;  
 }  
 public Electronic(String name, double price, String brand) {  
 super(name, price);  
 this.brand = brand;  
 }  
  
 @Override  
 public void printDetails() {  
 System.*out*.println("Product Name: " + name);  
 System.*out*.println("Brand: " + brand);  
 System.*out*.println("Price: " + price);  
 System.*out*.println("Discounted Price: " + getDiscountedPrice());  
 }  
  
}  
class Clothing extends Product{  
 private String size;  
 public double getDiscountedPrice() {  
 return price \*0.15 ;  
 }  
 public Clothing(String name, double price, String size) {  
 super(name, price);  
 this.size = size;  
 }  
 @Override  
 public void printDetails() {  
 System.*out*.println("Product Name: " + name);  
 System.*out*.println("Size: " + size);  
 System.*out*.println("Price: " + price);  
 System.*out*.println("Discounted Price: " + getDiscountedPrice());  
 }  
  
}  
interface Customer{  
 String *name* = "22k-5195";  
 double *balance* = 0;  
 abstract public void buyProduct(Product p);  
 abstract void printDetails();  
}  
class Customer1 implements Customer{  
 String name ;  
 double balance ;  
  
 public Customer1(String name, double balance) {  
 this.name = name;  
 this.balance = balance;  
 }  
  
 public void buyProduct(Product p){  
 double discount = p.getDiscountedPrice();  
 if (balance>discount){  
 balance -= discount;  
 System.*out*.println(name + " has bought the product: " + p.name);  
 }  
 else{  
 System.*out*.println(name + " can't buy the product: " + p.name);  
 }  
 }  
 public void printDetails() {  
 System.*out*.println("Name : " + name);  
 System.*out*.println("Balance : $" + balance);  
 }  
}  
interface VIPCustomer extends Customer {  
 @Override  
 default void buyProduct(Product product) {  
 double discountedPrice = product.getDiscountedPrice();  
 double additionalDiscount = 0.1 \* discountedPrice;  
 discountedPrice -= additionalDiscount;  
  
 if (*balance* >= discountedPrice) {  
  
 System.*out*.println(*name* + " (VIP) has bought the product: " + product.name);  
 }  
 else {  
 System.*out*.println(*name* + " (VIP) does not have enough balance to buy the product: " + product.name);  
 }  
 }  
  
}  
class VIPCustomerImpl implements VIPCustomer {  
 private String name;  
 private double balance;  
  
 public VIPCustomerImpl(String name, double balance) {  
 this.name = name;  
 this.balance = balance;  
 }  
  
 @Override  
 public void buyProduct(Product product) {  
 double discountedPrice = product.getDiscountedPrice() \* 0.9; // 10% additional discount for VIP customers  
 if (balance >= discountedPrice) {  
 balance -= discountedPrice;  
 System.*out*.println(name + " (VIP) has bought " + product.name + " for $" + discountedPrice);  
 } else {  
 System.*out*.println(name + " (VIP) does not have enough balance to buy " + product.name);  
 }  
 }  
  
 public void printDetails() {  
 System.*out*.println("VIP Customer Name: " + name);  
 System.*out*.println("Balance: $" + balance);  
 }  
}  
  
public class A2\_Q3\_22k5195 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5195 Laiba Fatima");  
 Product book = new Book("Alchemist", 790.99, "Paul");  
 Product electronic = new Electronic("J7 Prime", 100.0, "Samsung");  
 Product clothing = new Clothing("Dress", 98.0, "4");  
  
 Customer customer1 = new Customer1("Laiba",987.0 );  
  
 System.*out*.println("Customer1 details:");  
  
 customer1.buyProduct(book);  
  
 customer1.buyProduct(electronic);  
  
 customer1.buyProduct(clothing);  
 customer1.printDetails();  
  
 Customer customer2 = new Customer1("fatima", 100.0);  
 System.*out*.println("\nCustomer2 details:");  
  
 customer2.buyProduct(book);  
  
 customer2.buyProduct(electronic);  
  
 customer2.buyProduct(clothing);  
 customer2.printDetails();  
  
 System.*out*.println("\nVIP: ");  
 VIPCustomer vipCustomer1 = new VIPCustomerImpl("noor", 200);  
 vipCustomer1.buyProduct(book);  
 vipCustomer1.buyProduct(electronic);  
 vipCustomer1.buyProduct(clothing);  
 vipCustomer1.printDetails();  
  
 System.*out*.println("\nProduct details: ");  
  
 book.printDetails();  
 electronic.printDetails();  
 clothing.printDetails();  
  
  
  
 //part b  
 System.*out*.println("\nAverage price");  
 Product p1 = new Book("Harry Potter", 40.6, "Laiba" );  
 Product p2 = new Book("Philosopher's Stone", 98.6, "Fatima" );  
 Product p3 = new Book() ;  
 p3.add(p1,p2);  
 System.*out*.println(p3.getName());  
 System.*out*.println(p3.getPrice());  
  
  
 }  
}

Output:





**Question Four.       Total Marks /10 (2 Marks Each)**

Imagine you have been hired by Binary Vibes to work on the gamming project. They have given a base class structure and have asked to complete the given requirements below:

|  |
| --- |
| **GameObject** |
| * Name : String * X : int * Y : int |
| +     GameObject(string, int, int)         +     draw() : Void |

Now, create three subclasses that inherit from the "GameObject" class: "Player," "Enemy," and "PowerUp." Health should be an additional property for the Player class, damage should be an additional property for the Enemy class, and effect should be an additional property for the PowerUp class.

1. Code the classes (GameObject, Player, Enemy & PowerUp). Each subclass should have constructors that initialize their properties, furthermore the attribute “Name” in “GameObject” class is constant and it should be initialized by your valid student ID. The format of valid student ID is XXK-XXXX, whereas “X” is the number. Additionally, if required, create accessors and mutators.
2. To incorporate the additional properties of the Player, Enemy, and PowerUp objects, override the draw() method in each subclass.
3. Create a new class called Game with a dynamic array of GameObject objects in it. Implement a functionality of DrawAll() in it. DrawAll() is a Game class method that calls the draw() method on each GameObject object while looping through the array.
4. In the main() function, create instances of the Player, Enemy, and PowerUp classes and add them to the array of GameObject objects in the Game class. Then call the drawAll() method on the Game object to draw all the objects in the array.
5. Use the equals() method for Player class for comparing the health of two players.

*Solution:*

//22k-5195 Laiba Fatima  
import java.util.ArrayList;  
class GameObject{  
 private final String Name ;  
 private int x;  
 private int y;  
  
 public GameObject( String name, int x, int y) {  
 this.Name ="22K-5195";  
 this.x = x;  
 this.y = y;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public int getX() {  
 return x;  
 }  
  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 public int getY() {  
 return y;  
 }  
  
 public void setY(int y) {  
 this.y = y;  
 }  
  
 public void draw(){  
 System.*out*.println("Drawing a GameObject at (" + getX() + ", " + getY() + ")");  
  
 }  
}  
class Player extends GameObject{  
 public int health;  
  
 public Player(String name, int x, int y, int health) {  
 super(name, x, y);  
 this.health = health;  
 }  
 public void draw(){  
 System.*out*.println("Player: ");  
 System.*out*.println(getName());  
 System.*out*.println("Drawing a GameObject at (" + getX() + ", " + getY() + ")");  
 health-=10;  
 System.*out*.println("Health: "+health);  
 }  
 public boolean equals(Player p) {  
  
 return this.health == p.health;  
 }  
}  
class Enemy extends GameObject{  
 public int damage;  
  
 public Enemy(String name, int x, int y, int damage) {  
 super(name, x, y);  
 this.damage = damage;  
 }  
 public void draw(){  
 System.*out*.println("Enemy: ");  
 System.*out*.println(getName());  
 System.*out*.println("Drawing a GameObject at (" + getX() + ", " + getY() + ")");  
 damage+=10;  
 System.*out*.println("Damage: "+damage);  
 }  
}  
class PowerUp extends GameObject{  
 public String effect;  
  
 public PowerUp(String name, int x, int y, String effect) {  
 super(name, x, y);  
 this.effect = effect;  
 }  
 public void draw(){  
 System.*out*.println("Power Up : ");  
 System.*out*.println(getName());  
 System.*out*.println("Drawing a GameObject at (" + getX() + ", " + getY() + ")");  
 System.*out*.println("Effect: "+effect);  
  
 }  
}  
class Game {  
 public ArrayList<GameObject> gameObjects = new ArrayList<GameObject>();  
  
 public void addObject(GameObject g) {  
 gameObjects.add(g);  
 }  
  
 public void drawAll() {  
 for (GameObject g : gameObjects) {  
  
 g.draw();  
 }  
 }  
}  
public class A2\_Q4\_22k5195 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5195 Laiba Fatima");  
 Player p1 = new Player("Laiba", 10, 20, 100);  
 Player p2 = new Player("Mustafa", 30, 40, 200);  
 Player p3 = new Player("Hani", 30, 40, 100);  
 Enemy e = new Enemy("Umair", 50, 60, 50);  
 PowerUp p = new PowerUp("Hafsah", 70, 80, "Increase speed");  
  
 Game game = new Game();  
 game.addObject(p1);  
 game.addObject(p2);  
 game.addObject(p3);  
 game.addObject(e);  
 game.addObject(p);  
  
 game.drawAll();  
  
  
 System.*out*.println(" Player 1 health equals Player 2 health? " + p1.equals(p2));  
 System.*out*.println(" Player 1 health equals Player 3 health? " + p1.equals(p3));  
  
 }  
}

Output:

