# OOP ASSIGNMENT : 02

24K-3055

Bse-2A

Malik Zaryab Awan

**Q1:**

package Q1;  
public class Bus {  
 private String busNumber;  
 private Route route;  
 private int capacity;  
  
 public Bus(String busNumber, int capacity) {  
 this.busNumber = busNumber;  
 this.capacity = capacity;  
 }  
  
 public void assignRoute(Route route) {  
 this.route = route;  
 System.*out*.println("Route assigned to bus " + busNumber);  
 }  
  
 public void checkUserTap(User user) {  
 if (user.tapCard()) {  
 System.*out*.println(user.name + " allowed on the bus.");  
 } else {  
 System.*out*.println(user.name + " denied entry.");  
 }  
 }  
}

package Q1;  
  
import java.util.\*;  
public class Route implements Comparable<Route> {  
 private int routeId;  
 private List<String> stops;  
  
 public Route(int routeId) {  
 this.routeId = routeId;  
 this.stops = new ArrayList<>();  
 }  
  
 public void addStop(String stop) {  
 stops.add(stop);  
 }  
  
 public void removeStop(String stop) {  
 stops.remove(stop);  
 }  
  
 @Override  
 public int compareTo(Route other) {  
 return Integer.*compare*(this.routeId, other.routeId);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Route route = (Route) obj;  
 return routeId == route.routeId && stops.equals(route.stops);  
 }  
}

package Q1;  
public class StaffMember extends User {  
 private String department;  
  
 public StaffMember(int id, String name, String cardNumber, double balance, String department) {  
 super(id, name, cardNumber, balance);  
 this.department = department;  
 }  
  
 @Override  
 public void payFees() {  
 processPayment(500);  
 }  
}

package Q1;  
public class Student extends User {  
  
 private double semesterBalance;  
  
 public Student(int id, String name, String cardNumber, double balance, double semesterBalance) {  
 super(id, name, cardNumber, balance);  
 this.semesterBalance = semesterBalance;  
 }  
  
 @Override  
 public void payFees() {  
 processPayment(semesterBalance);  
 }  
}

package Q1;  
public abstract class User {  
 protected int id;  
 protected String name;  
 protected String cardNumber;  
 protected double balance;  
  
 public User(int id, String name, String cardNumber, double balance) {  
 this.id = id;  
 this.name = name;  
 this.cardNumber = cardNumber;  
 this.balance = balance;  
 }  
  
 public abstract void payFees();  
  
 public boolean tapCard() {  
 if (balance >= 500) {  
 balance -= 500;  
 System.*out*.println(name + " tapped card. Entry granted.");  
 return true;  
 } else {  
 System.*out*.println(name + " has insufficient balance!");  
 return false;  
 }  
 }  
  
 public void processPayment(double amount) {  
 balance += amount;  
 System.*out*.println(name + " paid " + amount + ". New balance: " + balance);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 User user = (User) obj;  
 return id == user.id && name.equals(user.name);  
 }  
}

package Q1;  
public class Teacher extends User {  
 private double monthlySalary;  
  
 public Teacher(int id, String name, String cardNumber, double balance, double monthlySalary) {  
 super(id, name, cardNumber, balance);  
 this.monthlySalary = monthlySalary;  
 }  
  
 @Override  
 public void payFees() {  
 processPayment(monthlySalary);  
 }  
}

package Q1;  
public class Main {  
 public static void main(String[] args) {  
 User student1 = new Student(101, "Syed Arham", "CARD123", 600, 5000);  
 User teacher1 = new Teacher(201, "Xaryab", "CARD456", 1000, 10000);  
 User staff1 = new StaffMember(301, "Saad", "CARD789", 700, "Admin");  
 Route route1 = new Route(1);  
 route1.addStop("Stop A");  
 route1.addStop("Stop B");  
 Bus bus1 = new Bus("BUS-01", 50);  
 bus1.assignRoute(route1);  
 student1.payFees();  
 teacher1.payFees();  
 staff1.payFees();  
 bus1.checkUserTap(student1);  
 bus1.checkUserTap(teacher1);  
 bus1.checkUserTap(staff1);  
 Route route2 = new Route(1);  
 route2.addStop("Stop A");  
 route2.addStop("Stop B");  
 System.*out*.println("Are route1 & route2 the same? " + route1.equals(route2));  
 System.*out*.println("Are student1 & teacher1 the same? " + student1.equals(teacher1));  
 }  
}

**OUTPUT:**

****

**Q2:**

package Q2;  
  
public class Banshee extends Ghost{  
  
 public Banshee(String name, int scaryLevel) {  
 super(name, scaryLevel);  
 }  
  
 @Override  
 void ghostHaunts(){  
 super.ghostHaunts();  
 System.*out*.println(name + "Screams Loudly with a scare level of " + scaryLevel + "!");  
 }  
}

package Q2;  
  
public class Ghost {  
 protected String name;  
 protected int scaryLevel;  
  
 Ghost(String name, int scaryLevel) {  
 this.name = name;  
 if(scaryLevel>=0 && scaryLevel<=10) {  
 this.scaryLevel = scaryLevel;  
 }  
 else {  
 this.scaryLevel = 0;  
 }  
 }  
  
 void ghostHaunts() {  
 System.*out*.println("Ghost Starts Haunting");  
 }  
  
 void displayGhostInfo(){  
 System.*out*.println("Ghost Player's Name: " + this.name);  
 System.*out*.println("Ghost Scary Level: " + this.scaryLevel);  
 }  
  
 public void upgradeGhost(String secondName,int scaryLevel) {  
 this.name += " and " + secondName;  
 this.scaryLevel += scaryLevel;  
 System.*out*.println("Ghost upgraded! New scare level: " + this.scaryLevel);  
 }  
}

package Q2;  
import java.util.\*;  
  
import java.util.ArrayList;  
  
public class HauntedHouse {  
 private String name;  
 List<ShadowGhost> shadowGhost=new ArrayList<>();  
 List<Poltergeist> poltergeist=new ArrayList<>();  
 List<Banshee> banshee=new ArrayList<>();  
  
 public HauntedHouse(String name) {  
 this.name = name;  
 }  
  
 void addGhost(ShadowGhost ghost) {  
 shadowGhost.add(ghost);  
 }  
  
 void addGhost(Poltergeist ghost) {  
 poltergeist.add(ghost);  
 }  
  
 void addGhost(Banshee ghost) {  
 banshee.add(ghost);  
 }  
}

package Q2;  
  
public class Poltergeist extends Ghost{  
  
  
 public Poltergeist(String name, int scaryLevel) {  
 super(name, scaryLevel);  
 }  
  
 @Override  
 void ghostHaunts(){  
 super.ghostHaunts();  
 System.*out*.println(name + " moving objects with a scare level of " + scaryLevel + "!");  
 }  
}

package Q2;  
  
public class ShadowGhost extends Ghost {  
  
 public ShadowGhost(String name, int scaryLevel) {  
 super(name, scaryLevel);  
 }  
  
 @Override  
 void ghostHaunts(){  
 super.ghostHaunts();  
 System.*out*.println(name + " Whispers Creepily with a scare level of " + scaryLevel + "!");  
 }  
}

package Q2;  
  
public class ShadowPoltergeist extends Ghost{  
  
 ShadowPoltergeist(String name,int scaryLevel){  
 super(name,scaryLevel);  
 }  
  
 @Override  
 void ghostHaunts(){  
 System.*out*.println(name + " is whispering creepily and moving objects with a scare level of " + scaryLevel + "!");  
 }  
}

package Q2;  
  
public class Visitor {  
 private String name;  
 private int bravery;  
  
 Visitor(String name, int bravery) {  
 this.name = name;  
 if(bravery>=0 && bravery<=10) {  
 this.bravery = bravery;  
 }  
 else {  
 this.bravery=0;  
 }  
 }  
}

package Q2;  
  
public class Main {  
 public static void main(String[] args) {  
  
 HauntedHouse DragonBallZ = new HauntedHouse("DragonBallZ");  
 HauntedHouse AttackOnTitan = new HauntedHouse("AttackOnTitan");  
  
  
 DragonBallZ.addGhost(new ShadowGhost("Goku", 10));  
 DragonBallZ.addGhost(new Poltergeist("Vegeta", 9));  
 DragonBallZ.addGhost(new Banshee("Frieza", 9));  
 DragonBallZ.addGhost(new ShadowGhost("Beerus", 11));  
  
  
 AttackOnTitan.addGhost(new Poltergeist("Eren", 8));  
 AttackOnTitan.addGhost(new Banshee("Armin", 4));  
 AttackOnTitan.addGhost(new ShadowGhost("Levi", 10));  
  
  
 Visitor[] visitors = {  
 new Visitor("Xaryab", 9),  
 new Visitor("Arham", 5),  
 new Visitor("Rafay", 2)  
 };  
  
 System.*out*.println("HAUNTED HOUSE SIMULATION");  
 System.*out*.println();  
  
  
 Ghost testGhost = new ShadowGhost("Test\_Ghost", 5);  
 testGhost.displayGhostInfo();  
 testGhost.ghostHaunts();  
 System.*out*.println();  
  
  
 testGhost.upgradeGhost("Enhanced\_Ghost", 3);  
 testGhost.displayGhostInfo();  
 testGhost.ghostHaunts();  
 System.*out*.println();  
  
  
 for (ShadowGhost ghost : DragonBallZ.shadowGhost) {  
 ghost.displayGhostInfo();  
 }  
 for (Poltergeist ghost : DragonBallZ.poltergeist) {  
 ghost.displayGhostInfo();  
 }  
 for (Banshee ghost : DragonBallZ.banshee) {  
 ghost.displayGhostInfo();  
 }  
 System.*out*.println();  
  
 System.*out*.println("Haunting in progress...");  
 for (ShadowGhost ghost : AttackOnTitan.shadowGhost) {  
 ghost.ghostHaunts();  
 }  
 for (Poltergeist ghost : AttackOnTitan.poltergeist) {  
 ghost.ghostHaunts();  
 }  
 for (Banshee ghost : AttackOnTitan.banshee) {  
 ghost.ghostHaunts();  
 }  
 }  
}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Q3:**

package Q3;  
  
public class RamzanDrone extends Transport {  
  
 public RamzanDrone(int transportID, double velocity, int loadCapacity, double fuelEfficiency) {  
 super(transportID, velocity, loadCapacity, fuelEfficiency);  
 }  
  
 @Override  
 void calculateOptimizedRoute() {  
 System.*out*.println("Calculating aerial route for high-speed delivery...");  
 }  
  
 @Override  
 void estimateDeliveryTime(double velocity, double distance) {  
 distance = distance \* 1.82; // Nautical miles aerial distance  
 double time = distance / velocity;  
 System.*out*.println("Estimated Delivery Time for Ramzan Drone: " + time + " s");  
 }  
  
 @Override  
 void executeCommand(String action, int packageID, String priorityLevel) {  
 if (priorityLevel.equalsIgnoreCase("urgent")) {  
 System.*out*.println("Activating high-speed delivery mode...");  
 }  
 }  
}

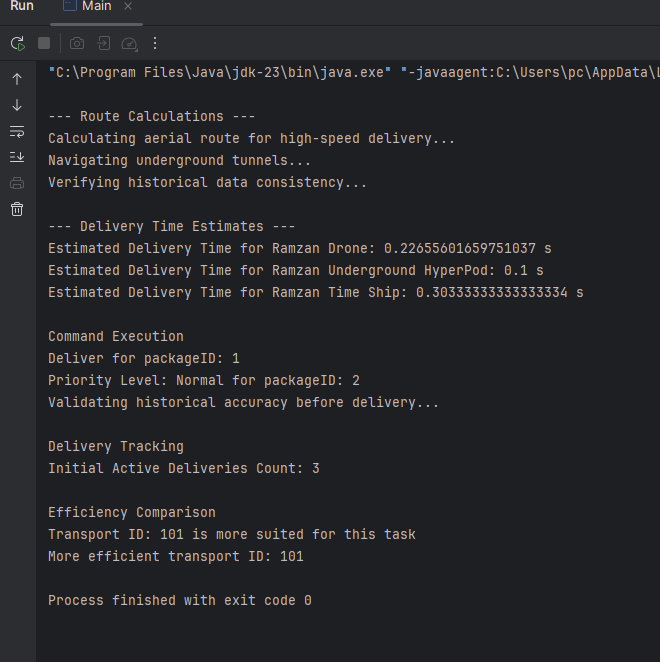
package Q3;  
public class RamzanHyperPod extends Transport {  
  
 public RamzanHyperPod(int transportID, double velocity, int loadCapacity, double fuelEfficiency) {  
 super(transportID, velocity, loadCapacity, fuelEfficiency);  
 }  
  
 @Override  
 void calculateOptimizedRoute() {  
 System.*out*.println("Navigating underground tunnels...");  
 }  
  
 @Override  
 void estimateDeliveryTime(double velocity, double distance) {  
 double time = distance / velocity;  
 System.*out*.println("Estimated Delivery Time for Ramzan Underground HyperPod: " + time + " s");  
 }  
}

package Q3;  
  
public class RamzanTimeShip extends Transport{  
 public RamzanTimeShip(int transportID, double velocity, int loadCapacity, double fuelEfficiency) {  
 super(transportID, velocity, loadCapacity, fuelEfficiency);  
 }  
  
 @Override  
 void calculateOptimizedRoute() {  
 System.*out*.println("Verifying historical data consistency...");  
 }  
  
 @Override  
 void estimateDeliveryTime(double velocity, double distance) {  
 distance = distance \* 1.82; // Nautical miles aerial distance  
 double time = distance / velocity;  
 System.*out*.println("Estimated Delivery Time for Ramzan Time Ship: " + time + " s");  
 }  
  
 @Override  
 void executeCommand(String action, int packageID, String priorityLevel) {  
 if (priorityLevel.equalsIgnoreCase("urgent")) {  
 System.*out*.println("Validating historical accuracy before delivery...");  
 }  
 }  
}

package Q3;  
  
public class Transport {  
 protected int transportID;  
 protected double velocity;  
 protected int loadCapacity;  
 protected double fuelEfficiency;  
 protected static int *activeDeliveriesCount*;  
  
 Transport(int transportID, double velocity, int loadCapacity, double fuelEfficiency) {  
 this.transportID = transportID;  
 this.velocity = velocity;  
 this.loadCapacity = loadCapacity;  
 this.fuelEfficiency = fuelEfficiency;  
 *activeDeliveriesCount*++;  
 }  
  
 void calculateOptimizedRoute() {  
 System.*out*.println("Calculating default route...");  
 }  
  
 void executeCommand(String action, int packageID) {  
 System.*out*.println(action + " for packageID: " + packageID);  
 }  
  
 void executeCommand(String action, int packageID, String priorityLevel) {  
 System.*out*.println("Priority Level: " + priorityLevel + " for packageID: " + packageID);  
 }  
  
 void estimateDeliveryTime(double velocity, double distance) {  
 double time = distance / velocity;  
 System.*out*.println("Estimated Time: " + time + " s");  
 }  
  
 void getActiveDeliveriesCount() {  
 System.*out*.println("Active Deliveries Count: " + *activeDeliveriesCount*);  
 }  
  
 static Transport compareFuelEfficiency(Transport t1, Transport t2) {  
 if (t1.fuelEfficiency > t2.fuelEfficiency) {  
 System.*out*.println("Transport ID: " + t1.transportID + " is more suited for this task");  
 return t1;  
 } else {  
 System.*out*.println("Transport ID: " + t2.transportID + " is suited for this task");  
 return t2;  
 }  
 }  
  
 void setTransportID(int transportID) {  
 this.transportID = transportID;  
 }  
  
 void setVelocity(double velocity) {  
 this.velocity = velocity;  
 }  
  
 void setLoadCapacity(int loadCapacity) {  
 this.loadCapacity = loadCapacity;  
 }  
  
 void setFuelEfficiency(double fuelEfficiency) {  
 this.fuelEfficiency = fuelEfficiency;  
 }  
}

package Q3;  
  
public class Main {  
 public static void main(String[] args) {  
 RamzanDrone drone = new RamzanDrone(101, 120.5, 5, 0.8);  
 RamzanHyperPod hyperPod = new RamzanHyperPod(201, 300.0, 50, 0.6);  
 RamzanTimeShip timeShip = new RamzanTimeShip(301, 150.0, 20, 0.7);  
  
 System.*out*.println("\n--- Route Calculations ---");  
 drone.calculateOptimizedRoute();  
 hyperPod.calculateOptimizedRoute();  
 timeShip.calculateOptimizedRoute();  
  
 System.*out*.println("\n--- Delivery Time Estimates ---");  
 drone.estimateDeliveryTime(120.5, 15.0);  
 hyperPod.estimateDeliveryTime(300.0, 30.0);  
 timeShip.estimateDeliveryTime(150.0, 25.0);  
  
 System.*out*.println("\nCommand Execution ");  
 drone.executeCommand("Deliver", 1);  
 hyperPod.executeCommand("Deliver", 2, "Normal");  
 timeShip.executeCommand("Deliver", 3, "Urgent");  
  
 System.*out*.println("\nDelivery Tracking ");  
 System.*out*.print("Initial ");  
 drone.getActiveDeliveriesCount();  
  
 System.*out*.println("\nEfficiency Comparison ");  
 Transport efficientTransport = Transport.*compareFuelEfficiency*(drone, hyperPod);  
 System.*out*.println("More efficient transport ID: " + efficientTransport.transportID);  
 }  
}

**OUTPUT:**

****

**Q4:**

package Q4;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public class User {  
 protected String name;  
 protected int id;  
 protected ArrayList<String> List\_of\_permissions;  
 protected String email;  
 protected int hashedPassword;  
  
 public int calculateHashedPassword(String password) {  
 int hash=5381;  
 for(char ch : password.toCharArray()) {  
 hash=hash\*33+ch;  
 }  
 return hash;  
 }  
 boolean authenticate(String password) {  
 if(hashedPassword==calculateHashedPassword(password)) {  
 return true;  
 }  
 else{  
 return false;  
 }  
 }  
 public User(String name, int id, ArrayList<String> list\_of\_permissions, String email, String password) {  
 this.name = name;  
 this.id = id;  
 this.List\_of\_permissions = list\_of\_permissions;  
 this.email = email;  
 this.hashedPassword = calculateHashedPassword(password);  
 }  
 public void Display\_details() {  
 System.*out*.println("Name: " + name);  
 System.*out*.println("ID: " + id);  
 System.*out*.println("List of permissions: " + List\_of\_permissions);  
 System.*out*.println("Email: " + email);  
 System.*out*.println("Password: " + hashedPassword);  
 }  
 public void Access\_lab(){  
 if(Arrays.*asList*(List\_of\_permissions).contains("LabAccess")) {  
 System.*out*.println("Access lab access");  
 }  
 else {  
 System.*out*.println("Access denied");  
 }  
 }  
}

package Q4;  
  
import java.util.ArrayList;  
  
public class TA extends User{  
 ArrayList<Student> students;  
 ArrayList<String> projects;  
  
 public TA(String name, int id, ArrayList<String> list\_of\_permissions, String email, String password, ArrayList<Student> assignStudents) {  
 super(name, id, list\_of\_permissions, email, password);  
 this.students = assignStudents;  
 projects = new ArrayList<>();  
 }  
  
 @Override  
 public void Display\_details() {  
 super.Display\_details();  
 }  
 void assignStudent(Student s){  
 if(students.size()>=10){  
 System.*out*.println("Student limit reached");  
 }  
 else{  
 students.add(s);  
 }  
 }  
 public void workWithProfessor(String project){  
 if(this.projects.size()>=2){  
 System.*out*.println("Project limit reached");  
 }  
 else{  
 this.projects.add(project);  
 System.*out*.println("Started Working on Project");  
 }  
 }  
}

package Q4;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Student extends User{  
 ArrayList<String> Assignments;  
 public Student(String name, int id, ArrayList<String> list\_of\_permissions, String email, String password) {  
 super(name, id, list\_of\_permissions, email, password);  
 Assignments=new ArrayList<>();  
 }  
  
 @Override  
 public void Display\_details() {  
 System.*out*.println("Student Details");  
 super.Display\_details();  
 }  
 public void AddAssignment(String assignment) {  
 Assignments.add(assignment);  
 }  
 public void updateAssignment(ArrayList<String> assignment) {  
 System.*out*.println("1 for assignment sumbitted\n0 for assingment left\n-1 to exit");  
 int counter=0;  
 for(String s:Assignments){  
 counter++;  
 System.*out*.println("Assignemnt "+counter+": ");  
 Scanner sc=new Scanner(System.*in*);  
 String answer=sc.nextLine();  
 if(answer.equals("-1")){  
 break;  
 }  
 else{  
 assignment.add(answer);  
 }  
 }  
 }  
 public void displayAssignment() {  
 System.*out*.println("Assignments---> 1 submitted, 0---->Left");  
 for(String s:Assignments){  
 System.*out*.println(s);  
 }  
 }  
  
}

package Q4;  
  
import java.util.ArrayList;  
  
public class Professor extends User{  
 TA ta;  
  
 public Professor(String name, int id, ArrayList<String> list\_of\_permissions, String email, String password) {  
 super(name, id, list\_of\_permissions, email, password);  
 }  
 @Override  
 public void Display\_details(){  
 System.*out*.println("Professor Details:");  
 super.Display\_details();  
 }  
 public void workOnTAProjects(TA ta) {  
 this.ta = ta;  
 System.*out*.println("TA Added");  
 System.*out*.println("Started to Work with TA's on Projects");  
 }  
  
 public void assignProjects(String projectName, TA ta) {  
 if(ta.projects.size()>=2){  
 System.*out*.println("Project already assigned to TA");  
 }  
 else{  
 ta.projects.add(projectName);  
 System.*out*.println("Project Assingned to TA");  
 }  
 }  
}

package Q4;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
 ArrayList<String> studentPermissions = new ArrayList<>(Arrays.*asList*("SubmitAssignment"));  
 ArrayList<String> taPermissions = new ArrayList<>(Arrays.*asList*("ViewProjects", "ManageStudents"));  
 ArrayList<String> profPermissions = new ArrayList<>(Arrays.*asList*("AssignProjects", "accessLab"));  
  
  
 ArrayList<String> assignments = new ArrayList<>(Arrays.*asList*("Math ", "Science", "Essay"));  
 Student S1=new Student("Xaryab",3055,studentPermissions,"k243055@nu.edu.pk","xaryab");  
 TA t1=new TA("Arham",2551,taPermissions,"k242551@nu.edu.pk","arham",new ArrayList<>());  
 Professor P1=new Professor("Rafay",3007,profPermissions,"k243007@nu.edu.pk","rafay");  
  
  
 System.*out*.println("tudent Operations ");  
 S1.Display\_details();  
 S1.displayAssignment();  
  
  
 System.*out*.println("\nTA Operations ");  
 t1.Display\_details();  
 t1.assignStudent(S1);  
 t1.workWithProfessor("1");  
  
 System.*out*.println("\n Professor Operations");  
 P1.Display\_details();  
 P1.workOnTAProjects(t1);  
 P1.assignProjects("Research Project", t1);  
  
 System.*out*.println("\n Authentication Test ");  
 System.*out*.println("Student authentication (correct password): " + S1.authenticate("student123"));  
 System.*out*.println("Student authentication (wrong password): " + S1.authenticate("wrongpass"));  
  
 System.*out*.println("\n===== Lab Access Test =====");  
 System.*out*.print("Student lab access: ");  
 S1.Access\_lab();  
 System.*out*.print("TA lab access: ");  
 t1.Access\_lab();  
 }  
}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.