**Main**

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Train train1 = new Train("Talgo","Astana -- Atbasar",500,35,140);  
 Train train2 = new Train("Compartment","Aktobe -- Astana",1700,30,85);  
 Train train3 = new Train("Electric","Almaty -- Astana",1250,45,120);  
 for (int i = 1; i <= 5; i++) {  
 Luxury luxury = new Luxury("Lux" + i, 20);  
 train1.addWagon(luxury);  
 train2.addWagon(luxury);  
 train3.addWagon(luxury);  
 Coupe coupe = new Coupe("Cou" + i, 30);  
 train1.addWagon(coupe);  
 train2.addWagon(coupe);  
 train3.addWagon(coupe);  
 Couchette couchette = new Couchette("Couch" + i, 60);  
 train1.addWagon(couchette);  
 train2.addWagon(couchette);  
 train3.addWagon(couchette);  
 }  
  
  
 while (true) {  
 System.*out*.println("---- Operations Management ----");  
 System.*out*.println("1. Add Wagon.");  
 System.*out*.println("2. Remove Wagon.");  
 System.*out*.println("3. Check Train Information.");  
 System.*out*.println("4. Exit.");  
 System.*out*.println("5. Check capacities of each type of Wagons.");  
 System.*out*.println("6. Check the Traction of the Locomotive.");  
 System.*out*.print("Enter your choice: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 if( choice == 4){  
 break;  
 }  
  
 System.*out*.println("Select Train (1 or 2 or 3): ");  
 int trainChoice = scanner.nextInt();  
 scanner.nextLine();  
  
 Train selectedTrain;  
 if (trainChoice == 1) {  
 selectedTrain = train1;  
 } else if (trainChoice == 2){  
 selectedTrain = train2;  
 }  
 else{  
 selectedTrain = train3;  
 }  
  
 if (choice == 1) {  
 System.*out*.print("Enter Wagon Type (Luxury - 1, Coupe - 2, Couchette - 3): ");  
 String type = scanner.nextLine();  
 System.*out*.print("Enter Wagon ID: ");  
 String id = scanner.nextLine();  
 switch (type) {  
 case "1":  
 Luxury luxury = new Luxury(id, 20);  
 selectedTrain.addWagon(luxury);  
 break;  
 case "2":  
 Coupe coupe = new Coupe(id, 30);  
 selectedTrain.addWagon(coupe);  
 break;  
 case "3":  
 Couchette couchette = new Couchette(id, 60);  
 selectedTrain.addWagon(couchette);  
 break;  
 }  
 } else if (choice == 2) {  
 System.*out*.print("Enter Wagon ID: ");  
 String id = scanner.nextLine();  
 for (Wagon wagon : selectedTrain.getWagons()) {  
 if (wagon.getId().equals(id)) {  
 selectedTrain.removeWagon(wagon);  
 break;  
 }  
 }  
 } else if (choice == 3) {  
 System.*out*.println("Total Train Capacity: " + selectedTrain.getTotalCapacity());  
 System.*out*.println("Train Speed: " + selectedTrain.getSpeed() + "km/h");  
 int distance = selectedTrain.getDistance();  
 System.*out*.println(selectedTrain.getName());  
 System.*out*.println("Travel Time: " + selectedTrain.getTravelTime(distance) + " hours");  
 System.*out*.println("Wagon Types and Counts:");  
 System.*out*.println("Luxury: " + selectedTrain.getWagonCount("Luxury"));  
 System.*out*.println("Coupe: " + selectedTrain.getWagonCount("Coupe"));  
 System.*out*.println("Couchette: " + selectedTrain.getWagonCount("Couchette"));  
 } else if (choice == 5){  
 System.*out*.println("Luxury - 20");  
 System.*out*.println("Coupe - 30.");  
 System.*out*.println("Couchette - 60.");  
 } else if (choice == 6){  
 System.*out*.println("Traction of the locomotive of the train "+selectedTrain.getName()+" is "+selectedTrain.getCarryingCapacity());  
 }  
 else {  
 System.*out*.println("Invalid Choice");  
 }  
 }  
 }  
}

**Train**

import java.util.List;  
import java.util.ArrayList;  
  
public class Train extends Locomotive {  
 private String locomotive;  
 private List<Wagon> wagons;  
 private String name;  
 private int distance;  
  
 public Train(String locomotive, String name, int distance, int carryingCapacity, int speed) {  
 super(speed,carryingCapacity);  
 setLocomotive(locomotive);  
 setWagons(new ArrayList<>());  
 setName(name);  
 setDistance(distance);  
 }  
  
 public void setLocomotive(String locomotive) {  
 this.locomotive = locomotive;  
 }  
  
 @Override  
 public int getCarryingCapacity() {  
 return super.getCarryingCapacity();  
 }  
  
 public void setWagons(List<Wagon> wagons) {  
 this.wagons = wagons;  
 }  
  
 public void addWagon(Wagon wagon) {  
 if (wagons.size() < getCarryingCapacity()) {  
 wagons.add(wagon);  
 } else {  
 System.*out*.println("Train is at maximum capacity.");  
 }  
 }  
  
 public void removeWagon(Wagon wagon) {  
 wagons.remove(wagon);  
 }  
 public int getTotalCapacity() {  
 int totalCapacity = 0;  
 for (Wagon wagon : wagons) {  
 if (wagon instanceof Luxury) {  
 totalCapacity += ((Luxury) wagon).getCapacity();  
 } else if (wagon instanceof Coupe) {  
 totalCapacity += ((Coupe) wagon).getCapacity();  
 } else if (wagon instanceof Couchette) {  
 totalCapacity += ((Couchette) wagon).getCapacity();  
 }  
 }  
 return totalCapacity;  
 }  
  
 public int getWagonCount(String wagonType) {  
 int count = 0;  
 for (Wagon wagon : wagons) {  
 if (wagon.getClass().getSimpleName().equals(wagonType)) {  
 count++;  
 }  
 }  
 return count;  
 }  
  
 @Override  
 public int getSpeed() {  
 return super.getSpeed();  
 }  
  
 public void setDistance(int distance) {  
 this.distance = distance;  
 }  
  
 public int getDistance() {  
 return distance;  
 }  
  
 public int getTravelTime(int distance) {  
 return distance / getSpeed();  
 }  
 public List<Wagon> getWagons() {  
 return wagons;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

**Locomotive**

public class Locomotive {  
 private int speed;  
 private int carryingCapacity;  
  
 public Locomotive(int speed, int carryingCapacity){  
 setSpeed(speed);  
 setCarryingCapacity(carryingCapacity);  
 }  
 public void setSpeed(int speed) {  
 this.speed = speed;  
 }  
  
 public int getSpeed() {  
 return speed;  
 }  
  
 public void setCarryingCapacity(int carryingCapacity) {  
 this.carryingCapacity = carryingCapacity;  
 }  
  
 public int getCarryingCapacity() {  
 return carryingCapacity;  
 }  
}

**Coupe**

public class Coupe extends Wagon {  
 private int capacity = 20;  
  
 public Coupe(String id, int capacity) {  
 super(id);  
 this.capacity = capacity;  
 }  
  
 public int getCapacity() {  
 return capacity;  
 }  
}

**Couchette**

public class Couchette extends Wagon {  
 private int capacity = 50;  
  
 public Couchette(String id, int capacity) {  
 super(id);  
 this.capacity = capacity;  
 }  
  
 public int getCapacity() {  
 return capacity;  
 }  
}

**Luxury**

public class Luxury extends Wagon {  
 private int capacity = 10;  
  
 public Luxury(String id, int capacity) {  
 super(id);  
 this.capacity = capacity;  
 }  
  
 public int getCapacity() {  
 return capacity;  
 }  
}

**Wagon**

public abstract class Wagon {  
 private String id;  
  
 public Wagon(String id) {  
 this.id = id;  
 }  
  
 public String getId() {  
 return id;  
 }  
}