//Main

import java.util.\*;

public class Main {  
 public static void main(String[] args) {  
 ArrayList<Car> cars = Functionality.*load*();  
 if (cars == null ) {  
 cars = new ArrayList<>();  
 }  
 Functionality.*menu*(cars);  
  
 }  
}

//Functionality Class

import java.io.\*;  
import java.util.\*;  
  
public class Functionality {  
  
 static Scanner *scanner* = new Scanner(System.*in*); *//static scanner for further usage* static int getInt() { *//getting integer input function* int inty = 0;  
 while (true) {  
 try {  
 inty = *scanner*.nextInt();  
 *scanner*.nextLine();  
 } catch (Exception e) {  
 System.*out*.println("Invalid");  
 *scanner*.nextLine();  
 continue;  
 }  
 return inty;  
 }  
 }  
  
 static void errorMassege() { *//error massege displaying* System.*out*.println("Invalid input. Please enter a valid number.");  
 *scanner*.nextLine();  
 }  
  
 static String carMakeAdding() { *//add car make at least 1 char* System.*out*.println("Firstly, we need to know a make of your car! Please, enter make of the car");  
 String make = "";  
 while (true) {  
 make = *scanner*.nextLine().trim();  
 if (!make.equals("")) {  
 break;  
 } else {  
  
 System.*out*.println("Please,enter valid make at least one character");  
 }  
 }  
 return make;  
 }  
  
 static String carModelAdding() { *//add car model at least 1 char* System.*out*.println("Now, enter your model");  
 String model = "";  
 while (true) {  
 model = *scanner*.nextLine().trim();  
 if (!model.equals("")) {  
 break;  
 } else {  
 System.*out*.println("Please,enter valid model at least one character");  
  
 }  
 }  
 return model;  
 }  
  
 static int carYearAdding() { *//add car year functionality* int year = 0;  
 System.*out*.println("So on this stage we need the year of your car: ");  
 boolean yearValueValid = false;  
 while (!yearValueValid) {  
 try {  
 System.*out*.println("Enter year...");  
 year = *scanner*.nextInt();  
 *scanner*.nextLine();  
 if (year <= 1885 || year > Calendar.*getInstance*().get(Calendar.*YEAR*)) { *//check if year is between 1885 and current year* System.*out*.println("You must enter a value more than 1885 or less/equal to " + Calendar.*getInstance*().get(Calendar.*YEAR*));  
 } else {  
 yearValueValid = true;  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 return year;  
 }  
  
 static int carTypeAdding() {  
 int typeChoice = 0;  
 boolean validChoice = false;  
 while (!validChoice) {  
 try {  
 System.*out*.println("Choose type:\n 1. ICE Car \n 2. Electric Car");  
 typeChoice = *scanner*.nextInt(); *//consume  
 scanner*.nextLine(); *//and* if (typeChoice >= 1 && typeChoice <= 2) { *//check user choice* validChoice = true;  
 } else {  
 System.*out*.println("Please, type a valid number.");  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 return typeChoice;  
 }  
  
 static Car createCarByType(String make, String model, int year, int typeChoice) {  
 Car newCar = null;  
 switch (typeChoice) {  
 case 1:  
 newCar = new ICECar(make, model, year, CarType.*ICE*);*//ICE car creation* break;  
 case 2:  
 System.*out*.println("Enter the battary capacity for this car in kWh");  
 int batteryCapacity = 0;  
 while (true) {  
 batteryCapacity = *getInt*(); *// consuming battery capacity* if (batteryCapacity <= 0 || batteryCapacity > 1000) {  
 System.*out*.println("Please, enter the number in between 0 and 1000");  
 } else {  
 newCar = new ElectricCar(make, model, year, CarType.*ELECTRIC*, batteryCapacity);*// electric car creation* break;  
 }  
 }  
  
 }  
  
 return newCar;  
 }  
  
 static void addCreateCar(ArrayList<Car> cars) {  
 String make = *carMakeAdding*(); *// assign values for make,model,year and car type* String model = *carModelAdding*();  
 int year = *carYearAdding*();  
 int typeChoice = *carTypeAdding*();  
 Car newCar = *createCarByType*(make, model, year, typeChoice);  
 *addCar*(cars, newCar);  
 }  
  
 static void addCar(ArrayList<Car> cars, Car newCar) {  
 if (cars != null && newCar != null) {  
 cars.add(newCar); *//adding new car* System.*out*.println("Car has been added successfully");  
 }  
 }  
  
 static void displayCars(ArrayList<Car> cars) {  
 if (cars.isEmpty()) { *//check if car list is empty* System.*out*.println("There are no cars in the list");  
 return;  
 } else {  
  
 int carOrderInTheList = 1;  
 for (Car car : cars) {  
 if (car instanceof ElectricCar) { *// car list display* System.*out*.println(carOrderInTheList + ". " + "[" + car.getMake() + ", " + car.getModel() + ", " + car.getCarType() + ", " + car.getYear() + ", " + ((ElectricCar) car).getBatteryCapacity() + "kWh" + "]");  
 carOrderInTheList++;  
 } else {  
 System.*out*.println(carOrderInTheList + ". " + "[" + car.getMake() + ", " + car.getModel() + ", " + car.getCarType() + ", " + car.getYear() + "]");  
 carOrderInTheList++;  
 }  
 }  
  
 }  
 }  
  
 static ArrayList<Car> load() { *// loading from file to ArrayList of cars* try {  
 FileInputStream fileInputStream = new FileInputStream("cars.ser");  
 ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
 ArrayList<Car> cars = (ArrayList<Car>) objectInputStream.readObject();  
 return cars;  
 } catch (IOException e) {  
 System.*out*.println("Something went wrong while loading a file or file has not been created!");  
 System.*out*.println(e.getMessage());  
 } catch (ClassNotFoundException e) {  
 System.*out*.println("Could not load a class from file.Source code probably has been changed.");  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
  
 static void save(ArrayList<Car> cars) { *//saving Array list to file* try {  
 FileOutputStream fileOutputStream = new FileOutputStream("cars.ser");  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(cars);  
 } catch (IOException e) {  
 System.*out*.println("Something went wrong while saving!");  
 }  
 }  
  
 static void removeCar(ArrayList<Car> cars) { *//remove car option* if (cars.isEmpty()) {  
 System.*out*.println("There are no cars to remove.");  
 return;  
 }  
 *displayCars*(cars);  
 boolean numberIsEntered = false;  
 while (!numberIsEntered) {  
 try {  
 System.*out*.println("Enter the number of the car you wish to remove:");  
 int carNum = *scanner*.nextInt();  
 *scanner*.nextLine();  
 if (carNum < 1 || carNum > cars.size()) {  
 System.*out*.println("Invalid car number of the car");  
 } else {  
 cars.remove(carNum - 1); *//removing car if it exists* System.*out*.println("Car removed successfully");  
 return;  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 }  
  
 static Car carForEditing(ArrayList<Car> cars) { *//defines what car will be edited  
 displayCars*(cars);  
 while (true) {  
 System.*out*.println("Enter the number of the car you wish to edit:");  
 try {  
 int carNum = *scanner*.nextInt();  
 *scanner*.nextLine();  
 if (carNum < 1 || carNum > cars.size()) {  
 System.*out*.println("Invalid car number. Please try again.");  
 } else {  
 return cars.get(carNum - 1); *//getting this car* }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 }  
  
 static void editMake(Car car) { *//editing make* String newMake = "";  
 System.*out*.println("Enter new make:");  
 while (true) {  
 newMake = *scanner*.nextLine();  
 if (!newMake.equals("")) {  
 car.setMake(newMake);  
 System.*out*.println("Make has been successfully updated");  
 break;  
 } else {  
 System.*out*.println("Please,try again!");  
 }  
  
 }  
 }  
  
 static void editModel(Car car) { *//editing model* String newModel="";  
 System.*out*.println("Enter new model:");  
 while (true) {  
 newModel = *scanner*.nextLine();  
 if (!newModel.equals("")) {  
 car.setModel(newModel);  
 System.*out*.println("Model has been updated successfully");  
 break;  
 } else {  
 System.*out*.println("Please, try again!");  
 }  
 }  
 }  
  
 static void editYear(Car car) { *//editing year* while (true) {  
 System.*out*.println("Enter new year:");  
 try {  
 int year = *scanner*.nextInt();  
 *scanner*.nextLine();  
 if (year <= 1885 || year > Calendar.*getInstance*().get(Calendar.*YEAR*)) {  
 System.*out*.println("Invalid year. Please enter a year more than 1885 and less or equal to the " + Calendar.*getInstance*().get(Calendar.*YEAR*));  
 } else {  
 car.setYear(year);  
 System.*out*.println("Year has been updated");  
 break;  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 }  
  
 static int displayEditOptionsAndUserInput() { *//Edit menu display* System.*out*.println("Choose an option: \n1. Make \n2. Model \n3. Year \n4. Exit to Menu ");  
 int input = *scanner*.nextInt();  
 *scanner*.nextLine();  
 return input;  
 }  
  
 static void editCar(ArrayList<Car> cars) { *//editing menu* if (cars.isEmpty()) {  
 System.*out*.println("There are no cars to edit");  
 return;  
 }  
 Car carToEdit = *carForEditing*(cars);  
 boolean exit = false;  
 while (!exit) {  
 try {  
 int choice = *displayEditOptionsAndUserInput*();  
 switch (choice) {  
 case 1:  
 *editMake*(carToEdit);  
 break;  
 case 2:  
 *editModel*(carToEdit);  
 break;  
 case 3:  
 *editYear*(carToEdit);  
 break;  
 case 4:  
 exit = true;  
 break;  
 default:  
 System.*out*.println("Invalid option. Please try again.");  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 }  
  
 static void menu(ArrayList<Car> cars) { *//main menu with 8 options to choose and option 9 to save automatically and exit* boolean exit = false;  
 while (!exit) {  
 *printMenu*();  
 try {  
 int choice = *scanner*.nextInt();  
 *scanner*.nextLine();  
 switch (choice) {  
 case 1:  
 *addCreateCar*(cars);  
 break;  
 case 2:  
 *displayCars*(cars);  
 break;  
 case 3:  
 *removeCar*(cars);  
 break;  
 case 4:  
 *editCar*(cars);  
 break;  
 case 5:  
 *carSorting*(cars);  
 break;  
 case 6:  
 *filterCars*(cars);  
 break;  
 case 7:  
 *avgYear*(cars);  
 break;  
 case 8:  
 *removeCarsByMake*(cars);  
 break;  
 case 9:  
 *save*(cars); *// Save cars before exiting* exit = true;  
 System.*out*.println("Exiting and saving data...");  
 break;  
 default:  
 System.*out*.println("Invalid option. Please try again.");  
 }  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 }  
  
 private static void printMenu() { *//print main menu text* System.*out*.println(  
 "Choose an option: \n" +  
 "1. Add Car \n" +  
 "2. View Cars \n" +  
 "3. Remove Car\n" +  
 "4. Edit Car\n" +  
 "5. Sort cars\n" +  
 "6. Filter by year\n" +  
 "7. AVG Car Year\n" +  
 "8. Remove by Make\n\n" +  
 "9. Exit ");  
 }  
  
 static void carSorting(ArrayList<Car> cars) {  
 ArrayList<Car> sortedCars = new ArrayList<>(cars);  
 if (sortedCars.isEmpty()) {  
 System.*out*.println("There no cars to sort");  
 return;  
 }  
 boolean exit = false;  
 while (!exit) {  
 System.*out*.println("Choose the sorting criteria: \n1. By Make \n2. By Year \n3. By CarType");  
 try {  
 int userInput = *scanner*.nextInt();  
 *scanner*.nextLine();  
 switch (userInput) {  
 case 1:  
 MakeComparator makeComparator = new MakeComparator(); *//comparator to sort by make* Collections.*sort*(sortedCars, makeComparator);  
 break;  
 case 2:  
 YearComparator yearComparator = new YearComparator(); *//comparator to sort by year* Collections.*sort*(sortedCars, yearComparator);  
 break;  
 case 3:  
 Collections.*sort*(sortedCars); *// compareTo enum sorting* break;  
 default:  
 System.*out*.println("Invalid option. Please try again.");  
 continue;  
 }  
 exit = true;  
 } catch (InputMismatchException e) {  
 *errorMassege*();  
 }  
 }  
 *displayCars*(sortedCars);  
 System.*out*.println("Cars have been sorted successfully.");  
 }  
  
 static ArrayList<Car> filterCars(ArrayList<Car> cars) { *//filtering car by year* ArrayList<Car> carsFiiltered = new ArrayList<>();  
 int yearToFilter = 0;  
 boolean numberIsValid = false;  
 while (!numberIsValid) {  
 *displayCars*(cars);  
 System.*out*.println("Enter the year to filter");  
 yearToFilter = *getInt*();  
 int year = Calendar.*getInstance*().get(Calendar.*YEAR*);  
 numberIsValid = 1886 < yearToFilter && yearToFilter <= year;  
 if (!numberIsValid) {  
 System.*out*.println("Invalid entry.Try again. Please enter the number between 1886 and" + year);  
 }  
 }  
 for (Car car : cars) { *//iterating through cars to add needed car a new array list* if (car.getYear() == yearToFilter) {  
 carsFiiltered.add(car);  
  
 }  
 }  
 *displayCars*(carsFiiltered); *// display filtered list of cars* return carsFiiltered;  
 }  
  
 static void avgYear(ArrayList<Car> cars) {  
 if (cars.isEmpty()) {  
 System.*out*.println("There is no cars in the list");  
 return;  
 }  
 double averageYear = 0;  
 for (Car car : cars) { *//calculating average year in the list by adding of years and devision by amount* averageYear += car.getYear();  
 }  
 averageYear = averageYear / cars.size();  
 System.*out*.println("The average year of cars is: " + Math.*round*(averageYear));  
 }  
  
 static void removeCarsByMake(ArrayList<Car> cars) {  
 if (cars.isEmpty()) {  
 System.*out*.println("No cars to remove");  
 return;  
 }  
 *displayCars*(cars);  
 System.*out*.println("Yo!What makes are you gonna remove?");  
 Iterator<Car> carIterator = cars.iterator();  
 String makeToRemove = *scanner*.nextLine().trim(); *//consuming make to remove* boolean removedCar = false;  
 while (carIterator.hasNext()) { *// iterator goes through list finding cars with needed make and removing them* Car carToRemove = carIterator.next();  
 if (carToRemove.getMake().toLowerCase().equals(makeToRemove.toLowerCase())) {  
 carIterator.remove();  
 removedCar = true;  
 }  
 }  
 if (removedCar == false) {  
 System.*out*.println("There is no cars with such a make");  
 } else {  
 System.*out*.println("All " + makeToRemove + " cars removed.");  
 }  
 }  
  
}

//Car Class

import java.io.Serializable;  
public class Car implements Serializable,Comparable<Car> {  
 private String make;  
 private String model; *//properties* private int year;  
 private final CarType carType;  
 public Car(String make, String model,int year,CarType carType) {  
 this.make = make;  
 this.model = model;  
 this.year = year;  
 this.carType = carType;  
  
 }  
 public String getMake() {  
 return make;  
 }  
 public void setMake(String make) {  
 this.make = make;  
 } *//setters and getters* public String getModel() {  
 return model;  
 }  
 public void setModel(String model) {  
 this.model = model;  
 }  
 public CarType getCarType() {  
 return carType;  
 }  
 public int getYear() {  
 return year;  
 }  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
  
 @Override  
 public int compareTo(Car other) {  
 return carType.compareTo(other.carType);  
 } *//compareTo to compare enum for sorting*}

//Car Test

import java.util.ArrayList;  
import static org.junit.jupiter.api.Assertions.\*;  
class CarTest {  
 ArrayList<Car>cars=new ArrayList<>();  
 @org.junit.jupiter.api.BeforeEach  
 void setUp() {  
 cars.add(new ICECar("Toyota", "CHR", 2022, CarType.*ICE*));  
 cars.add(new ElectricCar("Tesla", "ModelX", 2024, CarType.*ELECTRIC*,1000));  
 }  
 @org.junit.jupiter.api.AfterEach  
 void tearDown() {  
 }  
 @org.junit.jupiter.api.Test  
 void getMake() {  
 *assertEquals*("Toyota", cars.get(0).getMake());  
 *assertEquals*("Tesla", cars.get(1).getMake());  
 }  
 @org.junit.jupiter.api.Test  
 void setMake() {  
 cars.get(0).setMake("Not Toyota");  
 *assertEquals*("Not Toyota", cars.get(0).getMake());  
 cars.get(1).setMake("Not Tesla");  
 *assertEquals*("Not Tesla", cars.get(1).getMake());  
 }  
 @org.junit.jupiter.api.Test  
 void getModel() {  
 *assertEquals*("CHR", cars.get(0).getModel());  
 *assertEquals*("ModelX", cars.get(1).getModel());  
 }  
 @org.junit.jupiter.api.Test  
 void setModel() {  
 cars.get(0).setModel("Not CHR");  
 *assertEquals*("Not CHR", cars.get(0).getModel());  
 cars.get(1).setModel("Not ModelX");  
 *assertEquals*("Not ModelX", cars.get(1).getModel());  
 }  
 @org.junit.jupiter.api.Test  
 void getCarType() {  
 *assertEquals*(CarType.*ICE*, cars.get(0).getCarType());  
 *assertEquals*(CarType.*ELECTRIC*, cars.get(1).getCarType());  
 }  
 @org.junit.jupiter.api.Test  
 void getYear() {  
 *assertEquals*(2022, cars.get(0).getYear());  
 *assertEquals*(2024, cars.get(1).getYear());  
 }  
 @org.junit.jupiter.api.Test  
 void setYear() {  
 cars.get(0).setYear(2020);  
 *assertEquals*(2020, cars.get(0).getYear());  
 cars.get(1).setYear(2019);  
 *assertEquals*(2019, cars.get(1).getYear());  
 }  
}

//enum Car Type

public enum CarType {*//assuming two types of cars  
 ICE*,*ELECTRIC*}

//Make Comparator

import java.util.Comparator;  
  
public class MakeComparator implements Comparator<Car> {*//alphabet make comparator* @Override  
 public int compare(Car o1, Car o2) {  
 return o1.getMake().compareTo(o2.getMake());  
  
 }  
}

//Year Comparator

import java.util.Comparator;  
  
public class YearComparator implements Comparator<Car> {*//year comparator* @Override  
 public int compare(Car o1, Car o2) {  
 return o1.getYear() - o2.getYear();  
 }

//ICE Car extending from Car

public class ICECar extends Car {  
  
 public ICECar(String make, String model, int year, CarType carType) {  
 super(make, model, year, carType);  
 }  
}

//Electric Car extending from Car

import java.util.Date;  
  
public class ElectricCar extends Car{  
 final int batteryCapacity;  
 public ElectricCar(String make, String model, int year, CarType carType,int batteryCapacity) {  
 super(make, model, year, carType);  
 this.batteryCapacity = batteryCapacity;*// adding additional property to a subclass* ;  
 }  
  
 public int getBatteryCapacity() {  
 return batteryCapacity;  
 } *//getter for a new property*}