***class Car***

package com.company.car\_lambda;  
  
*/\*\*  
 \* Задание.  
 \* 1.Реализовать класс автомобиль, который содержит поля:  
 \* a.Марка автомобиля (компания и модель, например, toyota ipsum)  
 \* b.Стоимость автомобиля (руб)  
 \* c.Мощность автомобиля (л.с.)  
 \* 2.Добавить в класс все необходимые методы.  
 \* 3.Создать список автомобилей, заполнить его автомобилями с произвольными данными.  
 \* 4.Вывести данные об автомобилях на экран.  
 \* 5.Сделать выборку автомобилей ценой более 1 000 000 руб. Вывести данные на экран.  
 \* 6.Отсортировать список автомобилей по цене и мощности (автомобили с одной ценой \* сортировались по мощности). Вывести результат на экран.  
 \* 7.Найти автомобиль с указанной стоимостью  
 \* 8.Случилась инфляция, необходимо увеличить стоимость всех авто на 20%.  
 \* 9.Случилось снижение инфляции, необходимо снизить стоимость всех авто на 20%.  
 \* Примечание:  
 \* - Реализовать каждый пункт используя разные подходы:  
 \* - реализация объекта функционального интерфейса через класс  
 \* - реализация объекта функционального интерфейса через анонимный класс  
 \* - лямбда-выражения  
 \* при выполнении заданий минимально использовать цикл и императивный подход.  
 \*/*// класс "Автомобиль"  
public class Car {  
  
 // поля класса "Автомобиль"  
 private String nameModel; //название авто и модель  
 private double price; //Стоимость автомобиля  
 private int power; //мощность двигателя л.с.  
  
 // конструкторы  
 // 1. конструктор без параметров - по умолчанию  
 public Car() {  
 // инициализация полей по умолчанию  
 nameModel = " \"Model Car\"";  
 price = 1000000.0;  
 power = 80;  
  
 }  
  
 // 2. конструктор с 3 параметрами  
 public Car(String *nameModel*, double *price*, int *power*) {  
 this.nameModel = *nameModel*;  
 this.price = *price*;  
 this.power = *power*;  
  
 }  
  
 // getters and setters  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double *price*) {  
 this.price = *price*;  
 }  
  
 public int getPower() {  
 return power;  
 }  
  
 // 1. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString() {  
 return nameModel + ", мощность: " + power + " л.с., " + "цена " + price + " руб. ";  
 }  
}//class close

***interface Search***

package com.company.car\_lambda;  
  
import java.util.ArrayList;  
  
//Интерфейс - для поиска автомобилей по разным параметрам  
public interface Search {  
 void searchCarWithPrice(ArrayList<Car> *carList*, double *price*);  
}

***class Main***

package com.company.car\_lambda;  
  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.concurrent.atomic.AtomicInteger;  
  
public class Main {  
 public static void main(String[] *args*) {  
 // 1. Создание автомобилей  
 Car *rav4* = new Car("Toyota RAV4", 1325000, 158);  
 Car *camry* = new Car("Toyota Camry", 3500000, 200);  
 Car *civic* = new Car("Honda Civic", 730000, 140);  
 Car *accord* = new Car("Honda Accord", 1650000, 220);  
 Car *bmw\_x6* = new Car("BMW X6", 2000000, 555);  
 Car *mercedes\_g* = new Car("Mercedes-Benz G-Class", 4000000, 585);  
 Car *tucson* = new Car("Hyundai Tucson", 765000, 142);  
 Car *solaris* = new Car("Hyundai Solaris", 400000, 123);  
 Car *mpv* = new Car("Mazda MPV", 400000, 170);  
 Car *note* = new Car("Nissan Note", 400000, 109);  
  
 // 2. Создание списка автомобилей  
 ArrayList<Car> *carList* = new ArrayList<>();  
  
 // 3. Заполнение списка автомобилями  
 *carList*.add(*rav4*);  
 *carList*.add(*camry*);  
 *carList*.add(*civic*);  
 *carList*.add(*accord*);  
 *carList*.add(*bmw\_x6*);  
 *carList*.add(*mercedes\_g*);  
 *carList*.add(*tucson*);  
 *carList*.add(*solaris*);  
 *carList*.add(*mpv*);  
 *carList*.add(*note*);  
  
 // 4. Вывод данных об автомобилях на экран.  
 System.*out*.println("\t\t\t\tСписок автомобилей: ");  
 *carList*.forEach(System.*out*::println);  
  
 // 5. Выборка автомобилей ценой более 1 000 000 и вывод на экран  
 System.*out*.println("\n\t\t\t\tАвтомобили стоимостью больше 1 000 000 рублей: ");  
 *carList*.forEach((*car*) -> {  
 if (*car*.getPrice() > 1000000) {  
 System.*out*.println(*car*);  
 }  
 });  
  
 // 6. Сортировка автомобилей по цене и мощности (автомобили с одной ценой сортируются по мощности)  
 *carList*.sort(Comparator.*comparing*(Car::getPrice)  
 .thenComparing(Car::getPower));  
  
 System.*out*.println("\n\t\t\t\tАвтомобили отсортированные по цене и мощности: ");  
 // Вывод данных об автомобилях на экран.  
 *carList*.forEach(System.*out*::println);  
  
 // 7. Поиск автомобиля с указанной стоимостью  
 // (реализация объекта функционального интерфейса через анонимный класс)  
 System.*out*.println("\n\t\t\t\tПоиск автомобиля с указанной стоимостью: ");  
 Search *coastCar* = new Search() {  
 @Override  
 public void searchCarWithPrice(ArrayList<Car> *carList*, double *price*) {  
 // счетчик для выполнения условия если автомобиль не найден  
 AtomicInteger *count* = new AtomicInteger();  
 *carList*.forEach((*car*) -> {  
 if (*price* == *car*.getPrice()) {  
 System.*out*.println(*car*);  
 *count*.getAndIncrement();  
  
 }  
 });  
 if (*count*.get() == 0) {  
 System.*out*.println("Автомобиль с запрошенной ценой не найден");  
 }  
 }  
 };  
 *coastCar*.searchCarWithPrice(*carList*, 1325000.0);  
 *coastCar*.searchCarWithPrice(*carList*, 150000.0);  
  
 // 8. Случилась инфляция, необходимо увеличить стоимость всех авто на 20%  
 System.*out*.println("\n\t\t\t\tСлучилась инфляция, стоимость всех авто увеличилась на 20%: ");  
 *carList*.forEach((*car*) -> {  
 double *inflation* = (*car*.getPrice() \* 20 / 100) + *car*.getPrice();  
 *car*.setPrice(*inflation*);  
 System.*out*.println(*car*);  
 }  
 );  
  
 // 9. Случилось снижение инфляции, необходимо снизить стоимость всех авто на 20%.  
 System.*out*.println("\n\t\t\t\tСлучилась инфляция, стоимость всех авто снизилась на 20%: ");  
  
 *carList*.forEach((*car*) -> {  
 double *inflation* = *car*.getPrice() - (*car*.getPrice() \* 20 / 100);  
 *car*.setPrice(*inflation*);  
 System.*out*.println(*car*);  
 }  
 );  
  
 }//main  
}//class close

***Тест:***





