Proiect informatica

*“Parcare de autovehicule”*

Limbaj programare: Java

Realizat de:

Bretan Cezar

Jucan Rares

Profesor: Cristina Lupsa

12.06.2019

Overview problema:

Se considera o parcare care admite 4 tipuri de autovehicule: Sedan, TIR, ATV, Motocicleta. Aceste autovehicule se citesc din fisierul parcare.txt cu urmatorii parametri in ordinea respectiva:

* Initiala subclasei (String: S / T / A / M ; nu se va retine );
* An fabricatie (int);
* Brand (String);
* Model (String);
* Culoare (String: rosu / portocaliu / galben / verde / albastru / mov / argintiu / maro / negru / alb);
* Motor:
  + Configuratie (String: L / V / W / B );
  + Numar cilindrii (int);
  + Capacitate (float);
* Putere (int);
* [parametru specific subclasei]
* Masa (int);
* Valoare (float).

Fiecare subclasa are cate un parametru specific:

* Sedan: tipul tractiunii (String: FWD / RWD / AWD );
* TIR: masa remorcii (int);
* ATV: tip teren (String: Tarmac / Offroad);
* Motocicleta: clasa (String: Sport / Touring);

Ierarhie:



Citire:

Exemplu fisier:

**parcare.txt**

17

S 2007 VW Golf Argintiu L 4 1,3 165 FWD 1650 7500

S 2005 Opel Astra Negru L 4 1,6 140 RWD 1800 6000

S 1986 BMW M3 Albastru V 6 2,3 195 FWD 1500 7000

S 2010 Volvo S40 Alb L 5 2,5 227 AWD 1550 6500

S 2016 Mercedes-AMG GT-R Verde V 8 4,2 559 RWD 1900 80000

S 2003 Ford Focus Alb L 4 2,0 200 FWD 1400 6000

M 2009 Honda CBR1100 Rosu V 2 1,1 120 Sport 300 4500

M 2001 Harley-Davidson FXDR114 Negru V 2 1,8 114 Sport 450 7000

M 2018 Kawasaki NINJA650 Negru L 2 0,6 68 Touring 200 5630

M 2017 Ducati XDIAVEL Rosu V 2 1,2 156 Touring 220 21000

M 2003 Suzuki GSXR600 Mov L 4 0,6 100 Sport 200 6700

T 2014 Mercedes-Benz UnimogU5023 Argintiu V 8 7,0 230 1500 7000 100000

T 2016 Volvo IronKnight Negru W 12 9,0 2250 1200 4500 550000

T 2015 Iveco NewTrakker Verde V 10 7,5 450 2000 30000 180000

A 2016 Stels 800GGuepard Galben V 6 2,0 300 Offroad 850 8900

A 2018 Kawasaki BruteForce300 Negru L 1 0,3 21 Tarmac 600 4200

A 2019 Yamaha Grizzly350 Verde L 1 0,3 50 Offroad 120 6000

Pe prima linie se citeste numarul de autovehicule NrVehicule ce urmeaza sa fie citite. Pe urmatoarele NrVehicule linii se citesc datele in ordinea de mai sus. Prima litera are rolul de a indica carei subclase apartine obiectul caruia i se citesc parametrii de pe linia curenta.

int nrVehicule = sc.nextInt(); // Se citeste numarul de vehicule

ArrayList<Autovehicul> parcare = new ArrayList<>(nrVehicule); // Tablou care retine toate obiectele de tip Autovehicul, indiferent carei subclasa apartin

for (int i = 0; i < nrVehicule; i++) {

String tip\_curent = sc.next(); // Se citeste litera dinaintea autovehiculelor

switch (tip\_curent) { // Switch ce verifica care litera preceda autovehiculul curent

case ("S"): { // In cazul S, se instantiaza un Sedan si se introduce in tabloul parcare

// Citire parametri

int an\_fab = sc.nextInt();

String brand = sc.next();

String model = sc.next();

String culoare = sc.next();

Motor motor = new Motor();

motor.config = sc.next();

motor.nr\_cilindrii = sc.nextInt();

motor.capacitate = sc.nextFloat();

int putere = sc.nextInt();

String tractiune = sc.next(); // Parametru specific : tractiune

int masa = sc.nextInt();

float val = sc.nextFloat();

Sedan sedan\_curent = new Sedan(an\_fab, brand, model, culoare, motor, putere, tractiune, masa, val); // instantiere

parcare.add(sedan\_curent); // introducere in vector

break;

}

case ("T"): { // In cazul T, se instantiaza un TIR si se introduce in tabloul parcare

// Citire parametri

int an\_fab = sc.nextInt();

String brand = sc.next();

String model = sc.next();

String culoare = sc.next();

Motor motor = new Motor();

motor.config = sc.next();

motor.nr\_cilindrii = sc.nextInt();

motor.capacitate = sc.nextFloat();

int putere = sc.nextInt();

int masa\_rem = sc.nextInt(); // Parametru specific : masa remorcii

int masa = sc.nextInt();

float val = sc.nextFloat();

TIR tir\_curent = new TIR(an\_fab, brand, model, culoare, motor, putere, masa\_rem, masa, val); // instantiere

parcare.add(tir\_curent); // introducere in vector

break;

}

case ("A"): { // In cazul A, se instantiaza un ATV si se introduce in tabloul parcare

// Citire parametri

int an\_fab = sc.nextInt();

String brand = sc.next();

String model = sc.next();

String culoare = sc.next();

Motor motor = new Motor();

motor.config = sc.next();

motor.nr\_cilindrii = sc.nextInt();

motor.capacitate = sc.nextFloat();

int putere = sc.nextInt();

String tip\_teren = sc.next(); // Parametru specific : tip teren

int masa = sc.nextInt();

float val = sc.nextFloat();

ATV atv\_curent = new ATV(an\_fab, brand, model, culoare, motor, putere, tip\_teren, masa, val); // instantiere

parcare.add(atv\_curent); // introducere in vector

break;

}

case ("M"): { // In cazul M, se instantiaza o Motocicleta si se introduce in tabloul parcare

// Citire parametri

int an\_fab = sc.nextInt();

String brand = sc.next();

String model = sc.next();

String culoare = sc.next();

Motor motor = new Motor();

motor.config = sc.next();

motor.nr\_cilindrii = sc.nextInt();

motor.capacitate = sc.nextFloat();

int putere = sc.nextInt();

String clasa = sc.next(); // Parametru specific : clasa

int masa = sc.nextInt();

float val = sc.nextFloat();

Motocicleta moto\_curent = new Motocicleta(an\_fab, brand, model, culoare, motor, putere, clasa, masa, val); // instantiere

parcare.add(moto\_curent); // introducere in vector

break;

}

}

}

Interfata consola si functii:

**java.exe**

Bun venit in parcarea dumneavoastra. Pentru a incepe, apasati ENTER.

Ce ati dori sa aflati? Pentru a termina programul, scrieti 0. Pentru a curata consola, scrieti -1.

1) Toate autovehiculele din parcare.

2) Cea mai frecventa culoare.

3) Autovehiculele care au puterea motorului de cel putin x cai putere.

4) Autovehiculele de brand-ul x.

5) Autovehiculele ce se afla ca valoare intre doua preturi x si y in $.

6) Cate autovehicule au x roti.

7) Autovehiculele cu cel mult x ani vechime.

8) Autovehiculele cu masa cel mult x.

9) Autovehiculele au configuratia motorului x

10) Autovehiculele care au capacitatea cilindrica cel mult x

11) Numarul autovehiculelor de fiecare tip

12) Valoarea totala a autovehiculelor din parcare

13) Sedanurile cu tractiune de tip x

14) TIRurile cu masa remorcii cel mult x

15) ATVurile de teren de tip x

16) Motocicletele de clasa x

Inserati numar comanda:

Sursa Main.java contine functiile ce afiseaza in consola ceea ce este descris dupa numarul ce trebuie tastat pentru a o apela. Odata ce este tastat numarul functiei, se mai cere 1 sau 2 parametri (in cazurile 3 - 10 si 13 - 16) si se executa in mod direct functia aferenta numarului ei prin intermediul unui switch cu 18 cazuri (cazul -1 curata consola si reafiseaza interfata de mai sus, si 0 inchide programul).

Ex.:

Inserati numar comanda: 4

Brand: Volvo

4. 2010 Volvo S40

13. 2016 Volvo IronKnight

/// 1. Afiseaza toate autovehiculele

private static void toate\_autovehiculele(int n, ArrayList<Autovehicul> p){

for (int i = 0; i < n; i++) {

afisare(i, p);

}

}

/// 2. Afiseaza cea mai frecventa culoare din parcare si cate autovehicule o au

private static void cea\_mai\_folosita\_culoare(int n, ArrayList<Autovehicul> p) {

FrecvCuloare fr = new FrecvCuloare(); // Structura ce retine de cate ori apare fiecare culoare

for (int i = 0; i < n; i++) { // Se formeaza frecventa fiecarei culori

switch (p.get(i).getCuloare()) {

case ("Albastru"): {

fr.albastru++;

break;

}

case ("Argintiu"): {

fr.argintiu++;

break;

}

case ("Mov"): {

fr.mov++;

break;

}

case ("Galben"): {

fr.galben++;

break;

}

case ("Verde"): {

fr.verde++;

break;

}

case ("Negru"): {

fr.negru++;

break;

}

case ("Alb"): {

fr.alb++;

break;

}

case ("Portocaliu"): {

fr.portocaliu++;

break;

}

case ("Maro"): {

fr.maro++;

break;

}

case ("Rosu"): {

fr.rosu++;

break;

}

}

}

int contmax = 0;

String max = "";

// Se determina culoarea cea mai frecventa si frecventa ei

if(fr.portocaliu > contmax){

contmax = fr.portocaliu;

max = "Portocaliu";

}

if(fr.alb > contmax){

contmax = fr.alb;

max = "Alb";

}

if(fr.maro > contmax){

contmax = fr.maro;

max = "Maro";

}

if(fr.galben > contmax){

contmax = fr.galben;

max = "Galben";

}

if(fr.argintiu > contmax){

contmax = fr.argintiu;

max = "Argintiu";

}

if(fr.albastru > contmax){

contmax = fr.albastru;

max = "Albastru";

}

if(fr.mov > contmax){

contmax = fr.mov;

max = "Mov";

}

if(fr.negru > contmax){

contmax = fr.negru;

max = "Negru";

}

if(fr.rosu > contmax){

contmax = fr.rosu;

max = "Rosu";

}

if(fr.verde > contmax){

contmax = fr.verde;

max = "Verde";

}

System.out.print(contmax);

System.out.print(" autovehicule de culoare ");

System.out.print(max);

System.out.print('\n');

}

/// 3. Afiseaza autovehiculele cu puterea lor cel putin <cai> si puterea lor

private static void autovehicule\_putere\_min(int cai, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getPutere() >= cai) {

System.out.print(p.get(i).getPutere());

System.out.print(" CP - ");

afisare(i, p);

}

}

}

/// 4. Afiseaza autovehiculele de brand <br>

private static void autovehicule\_brand(String br, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getNumeBrand().equals(br)) afisare(i, p);

}

}

/// 5. Afiseaza autovehiculele cu valoarea cel putin <st> si cel mult <dr>

private static void autovehicule\_in\_price\_range(float st, float dr, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++) {

if(p.get(i).getValoare() <= dr && p.get(i).getValoare() >= st) {

System.out.print(p.get(i).getValoare());

System.out.print(" $ - ");

afisare(i, p);

}

}

}

/// 6. Afiseaza numarul autovehiculelor cu <r> roti

private static void nr\_autovehicule\_cu\_roti(int r, int n, ArrayList<Autovehicul> p){

int contor = 0;

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getNumarRoti() == r) contor++;

}

System.out.print(contor);

System.out.print(" autovehicule");

System.out.print('\n');

}

/// 7. Afiseaza autovehiculele de vechime cel mult <v>

private static void autovehicule\_de\_vechime\_max(int v, int n, ArrayList<Autovehicul> p) {

System.out.print('\n');

for(int i = 0; i < n; i++) {

if(2019 - p.get(i).getAnFabricatie() <= v) afisare(i, p);

}

}

/// 8. Afiseaza autovehiculele cu masa cel mult <m>

private static void autovehicule\_de\_masa\_max(int m, int n, ArrayList<Autovehicul> p) {

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getMasa() <= m){

System.out.print(p.get(i).getMasa());

System.out.print(" kg - ");

afisare(i, p);

}

}

}

/// 9. Afiseaza configuratia si nr de cilindrii ale autovehiculelor cu configuratia <c>

private static void autovehicule\_de\_config\_motor(String c, int n, ArrayList<Autovehicul> p){

for(int i = 0; i < n; i++){

if(p.get(i).getConfig().equals(c)){

System.out.print(p.get(i).getConfig());

System.out.print(p.get(i).getNrCilindrii());

System.out.print(" - ");

afisare(i, p);

}

}

}

/// 10. Afiseaza autovehiculele cu capacitatea cilindrica cel mult <c>

private static void autovehicule\_de\_capacit\_max(float c, int n, ArrayList<Autovehicul> p){

for(int i = 0; i < n; i++){

if(p.get(i).getCapacitate() <= c){

System.out.print(p.get(i).getCapacitate());

System.out.print(" L - ");

afisare(i, p);

}

}

}

/// 11. Afiseaza numarul sedanurilor, TIRurilor, ATVurilor si Motocicletelor

private static void nr\_autovehicule\_de\_fiecare\_tip(int n, ArrayList<Autovehicul> p){

int nrs = 0, nrt = 0, nra = 0, nrm = 0;

for(int i = 0; i < n; i++){

switch (p.get(i).getClass().toString()){ // Switch ce verifica carei subclase apartine obiectul de pe pozitia <i>

// Metoda predefinita .getClass() obtine subclasa careia apartine unui obiect de pe pozitia <i>,

// iar .toString() transpune denumirea acesteia in data de tip String sub forma "class [denumire\_subclasa]"

case("class Sedan"): { nrs++; break; }

case("class TIR"): { nrt++; break; }

case("class ATV"): { nra++; break; }

case("class Motocicleta"): { nrm++; break; }

}

}

System.out.print("Sedanuri: ");

System.out.print(nrs);

System.out.print('\n');

System.out.print("TIRuri: ");

System.out.print(nrt);

System.out.print('\n');

System.out.print("ATVuri: ");

System.out.print(nra);

System.out.print('\n');

System.out.print("Motociclete: ");

System.out.print(nrm);

System.out.print('\n');

}

/// 12. Afiseaza suma tuturor valorilor autovehiculelor

private static void valoare\_totala\_parcare(int n, ArrayList<Autovehicul> p){

long sum = 0;

for(int i = 0; i < n; i++){

sum += p.get(i).getValoare();

}

System.out.print("Valoarea totala a autovehiculelor din parcare: ");

System.out.print(sum);

System.out.print(" $");

System.out.print('\n');

}

/// 13. Afiseaza sedanurile cu tractiune de tip <tr>

private static void sedanuri\_cu\_tractiune(String tr, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getClass() == Sedan.class){

Sedan s = (Sedan) p.get(i);

if(s.getTractiune().equals(tr)) afisare(i, p);

}

}

}

/// 14. Afiseaza TIRurile cu masa remorcii cel mult <m>

private static void tiruri\_cu\_masa\_remorcii\_max(int m, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getClass() == TIR.class){

TIR t = (TIR) p.get(i);

System.out.print(t.getMasaRemorca());

System.out.print(" kg - ");

if(t.getMasaRemorca() <= m) afisare(i, p);

}

}

}

/// 15. Afiseaza ATVurile de teren de tip <t>

private static void atvuri\_de\_tip\_teren(String t, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getClass() == ATV.class){

ATV a = (ATV) p.get(i);

if(a.getTipTeren().equals(t)) afisare(i, p);

}

}

}

/// 16. Afiseaza motocicletele de clasa <c>

private static void motociclete\_de\_clasa(String c, int n, ArrayList<Autovehicul> p){

System.out.print('\n');

for(int i = 0; i < n; i++){

if(p.get(i).getClass() == Motocicleta.class){

Motocicleta m = (Motocicleta) p.get(i);

if(m.getClasa().equals(c)) afisare(i, p);

}

}

}

Clase:

**Autovehicul.java**

public abstract class Autovehicul implements AutovehiculImpl {

//PARAMETRII

private int an\_fabricatie;

private int putere;

private int masa;

protected int numar\_roti;

private float valoare;

private Motor motor;

private String brand;

private String model;

private String culoare;

//CONSTRUCTOR

Autovehicul(int an\_fabricatie, String brand, String model, String culoare, Motor motor, int putere, int masa, float valoare){

this.an\_fabricatie = an\_fabricatie;

this.brand = brand;

this.putere = putere;

this.masa = masa;

this.valoare = valoare;

this.model = model;

this.culoare = culoare;

this.motor = motor;

}

//GETTERS

\*Cate un getter pentru fiecare parametru\*

//SETTERS

\*Cate un setter pentru fiecare parametru\*

}

class Motor {

String config;

int nr\_cilindrii;

float capacitate;

}

class FrecvCuloare { // Structura ce retine de cate ori apare fiecare culoare

int rosu;

int portocaliu;

int galben;

int verde;

int albastru;

int mov;

int argintiu;

int maro;

int negru;

int alb;

}

**AutovehiculImpl.java**

public interface AutovehiculImpl {

int getAnFabricatie();

int getPutere();

int getMasa();

int getNumarRoti();

int getNrCilindrii();

float getValoare();

float getCapacitate();

String getConfig();

String getNumeBrand();

String getModel();

String getCuloare();

void setAnFabricatie(int an\_fabricatie);

void setPutere(int putere);

void setMasa(int masa);

void setValoare(float valoare);

void setConfig(String config);

void setNrCilindrii(int nr\_cilindrii);

void setCapacitate(float capacitate);

void setNumeBrand(String denumire);

void setModel(String model);

void setCuloare(String culoare);

}

**Sedan.java**

class Sedan extends Autovehicul implements SedanImpl {

private String tractiune;

Sedan(int an\_fabricatie, String brand, String model, String culoare, Motor motor, int putere, String tractiune, int masa, float valoare) {

super(an\_fabricatie, brand, model, culoare, motor, putere, masa, valoare);

this.numar\_roti = 4;

this.tractiune= tractiune;

}

public String getTractiune(){ return this.tractiune; }

public void setTractiune(String tractiune){ this.tractiune = tractiune; }

}

**TIR.java**

class TIR extends Autovehicul implements TIRImpl {

private int masa\_remorca;

TIR(int an\_fabricatie, String brand, String model, String culoare, Motor motor, int putere, int masa\_remorca, int masa, float valoare) {

super(an\_fabricatie, brand, model, culoare, motor, putere, masa, valoare);

this.numar\_roti = 6;

this.masa\_remorca = masa\_remorca;

}

public int getMasaRemorca(){ return this.masa\_remorca; }

public void setMasaRemorca(int masa\_remorca){ this.masa\_remorca = masa\_remorca; }

}

**ATV.java**

class ATV extends Autovehicul implements ATVImpl {

private String tip\_teren;

ATV(int an\_fabricatie, String brand, String model, String culoare, Motor motor, int putere, String tip\_teren, int masa, float valoare) {

super(an\_fabricatie, brand, model, culoare, motor, putere, masa, valoare);

this.numar\_roti = 4;

this.tip\_teren = tip\_teren;

}

public String getTipTeren(){ return this.tip\_teren; }

public void setTipTeren(String tip\_teren){ this.tip\_teren = tip\_teren; }

}

**Motocicleta.java**

class Motocicleta extends Autovehicul implements MotocicletaImpl {

private String clasa;

Motocicleta(int an\_fabricatie, String brand, String model, String culoare, Motor motor, int putere, String clasa, int masa, float valoare) {

super(an\_fabricatie, brand, model, culoare, motor, putere, masa, valoare);

this.numar\_roti = 2;

this.clasa = clasa;

}

public String getClasa(){ return this.clasa; }

public void setClasa(String clasa){ this.clasa = clasa; }

}

**SedanImpl.java**

public interface SedanImpl {

String getTractiune();

void setTractiune(String tractiune);

}

**TIRImpl.java**

public interface TIRImpl {

int getMasaRemorca();

void setMasaRemorca(int masa\_remorca);

}

**ATVImpl.java**

public interface ATVImpl {

String getTipTeren();

void setTipTeren(String tip\_teren);

}

**MotocicletaImpl.java**

public interface MotocicletaImpl {

String getClasa();

void setClasa(String clasa);

}