Darea de seamă la Java: Lucrare de laborator nr. 4

Tema: Colectii de obiecte in Java

Varianta: Nr. 16

Efectuat: Țurcanu Cristian, studentul grupei IA1901

Verificat: Epifanova Irina, lector universitar, magistru în informatică

Formularea problemei:

1. Pentru laboratorul Nr.1, de creat clasa adaugatoare, care descrie unitatea, setata in vectorul dinamic sau cateva vectori dinamici; (pentru Coctail -> class Ingredient, pentru Caravana -> class Camila, e t.c.)
2. Pentru clasa adaugata, de descris constructorul implicit cu randomizarea campurilor , constructor cu parametric si metoda de afisare a campurilor + set/get;
3. In clasa de baza, de inlocuit vectorul dynamic (sau cateva vectorii) cu ArrayList de tipul clasei adaugate;
4. De sters campul care arata nr. de elemente al vectorului dinamic;
5. De redescris (nu de sters) toate metodele clasei de baza, inlocuind prelucrarea vectorului dynamic cu prelucrarea ArrayList-ei.
6. Metoda statica main() de descris intr-o clasa aparte si de inlocuit vectorul dinamic de tip clasa de baza cu ArrayList.
7. In lucrare trebuie sa fie utilizat cicluri for si forEach.
8. De prezentat in IDE Intellij Idea

Codul sursă

CarteCopie.java

```
package com.company;

public class CarteCopie {
    private int state;
    private boolean hasAutograph;

    public CarteCopie() {
        state = Helper.randomizer.nextInt(10) + 1;
        hasAutograph = Helper.randomizer.nextInt(10) >= 7 ? true : false;
    }

    public CarteCopie(int state, boolean hasAutograph) {
        if (state < 1 || state > 10) {
            Helper.println("ERROR: incorrect value given to SetState: " + state +
", should be in [1,10]");
            throw new RuntimeException();
        }

        this.state = state;
        this.hasAutograph = hasAutograph;
```

```
    }

    public int getState() {
        return state;
    }

    public void setState(int state) {
        if (state < 1 || state > 10) {
            Helper.println("ERROR: incorrect value given to SetState: " + state +
", should be in [1,10]");
            throw new RuntimeException();
        }

        this.state = state;
    }

    public boolean getHasAutograph() {
        return hasAutograph;
    }

    public void setHasAutograph(boolean hasAutograph) {
        this.hasAutograph = hasAutograph;
    }

    public void Edit() {
        boolean success = false;
        do {
            try {
                Helper.println("Dati starea cartii: ");
                state = Helper.InputIntLimit(1, 10);
                Helper.println("Are autograf? (true/false): ");
                hasAutograph = Helper.InputBoolean();
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                success = false;
            }
        } while (!success);
    }

    public void Print() {
        String result = "Carte de starea " + state;

        if (hasAutograph)
            result += " cu autograful autorului";

        Helper.println(result);
    }
}
```

Carte.java

```
package com.company;

import java.util.*;

class Carte {
    private float pret;
    private String denumire;
    private String autor;

    private static int nrCarti = 0;
    private ArrayList<CarteCopie> state;

    public Carte() {
        pret = -1f;
        denumire = "---";
        autor = "---";

        nrCarti++;
        state = new ArrayList<CarteCopie>();
    }

    public Carte(float pret, String denumire, String autor, ArrayList<CarteCopie>
state) {
        this.pret = pret;
        this.denumire = denumire;
        this.autor = autor;
        nrCarti++;

        this.state = (ArrayList) state.clone();
    }

    public Carte(Carte copy) {
        this.pret = copy.getPret();
        this.denumire = copy.getDenumire();
        this.autor = copy.getAutor();
        nrCarti++;
        this.state = copy.getStateArrayList();
    }

    public float getPret() {
        return pret;
    }

    public void setPret(float pret) {
        if (pret < 0f) {
            System.out.println("ERROR: negative price given at setPret");
            throw new RuntimeException();
        }
        this.pret = pret;
    }

    public String getDenumire() {
        return denumire;
```

```
    }

    public void setDenumire(String denumire) {
        this.denumire = denumire;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public int getNrCopii() {
        return state.size();
    }

    public void setNrCopii(int nrCopii) {
        if (nrCopii < 0) {
            System.out.println("ERROR: negative nrCopii given");
            throw new RuntimeException();
        }

        if (nrCopii == this.state.size()) {
            System.out.println("WARNING: nrcopii == this.nrCopii, no changes");
            return;
        }

        for (int i = state.size(); i < nrCopii; i++) {
            state.add(new CarteCopie());
        }

        while (state.size() > nrCopii) {
            state.remove(state.size() - 1);
        }

//        int minCount = nrCopii < this.nrCopii ? nrCopii : this.nrCopii;
//
//        int temp[] = new int[this.nrCopii];
//
//        for (int i = 0; i < this.nrCopii; i++) {
//            temp[i] = state[i];
//        }
//
//        state = new int[nrCopii];
//        this.nrCopii = nrCopii;
//
//        for (int i = 0; i < minCount; i++) {
//            state[i] = temp[i];
//        }
    }

    public static int getNrCarti() {
```

```
        return nrCarti;
    }

    public CarteCopie getState(int id) {
        if (id >= 0 && id < state.size()) {
            return state.get(id);
        }

        System.out.println("Shit's fucked man");
        System.out.println("Requested invalid id at ***.getState");
        return null;
    }

    public void setState(int id, CarteCopie state) {
        if (id >= 0 && id < this.state.size()) {
            this.state.set(id, state);
        } else if (id < 0) {
            System.out.println("Set invalid id at ***.setState");
            throw new IndexOutOfBoundsException();
        } else {
            Helper.println("WARNING: set CarteCopie at too big position, adding to
end of list");
            this.state.add(state);
        }
    }

    public void addState(CarteCopie state) {
        this.state.add(state);
    }

    public ArrayList<CarteCopie> getStateArrayList() {
        return (ArrayList<CarteCopie>) state.clone();
    }

    public void setStateArrayList(ArrayList<CarteCopie> state) {
        this.state = (ArrayList<CarteCopie>)state.clone();
    }

    public void Print() {
        System.out.println("Pret: " + pret);
        System.out.println("Denumire: " + denumire);
        System.out.println("Autor: " + autor);
        System.out.println("Nr copii: " + state.size());
        System.out.println("Status: ");

        for (CarteCopie copie : state) {
            copie.Print();
        }

        System.out.println();
    }

    public void Edit() {
        System.out.print("Dati pretul: ");
```

```
        boolean success = false;
        do {
            try {
                setPret(Helper.InputFloat());
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                success = false;
            }
        } while (!success);

        System.out.print("Dati denumirea: ");
        setDenumire(Helper.InputString());

        System.out.print("Dati autorul: ");
        setAutor(Helper.InputString());

        System.out.print("Dati nr. copii: ");

        success = false;
        do {
            try {
                setNrCopii(Helper.InputInt());
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                success = false;
            }
        } while (!success);

        for (int i = 0; i < state.size(); i++) {
            setState(i, new CarteCopie());
            state.get(i).Edit();
        }
    }

    public void Randomize() {
        String titles[] = { "bandit of tomorrow", "lion of despair", "wives of the
mountain", "spies with strength",
                "armies and defenders", "traitors and girls", "fate of gold",
"root of the eclipse",
                "belonging to the king", "smiles in my family" };

        String names[] = { "Harper Gear", "Alex Robinson", "Brett Blade", "Franky
Joyce", "Caden Martin", "Jody Ramone",
                "Jude Day", "Caden Brandon", "Vic Money", "Quinn Davis" };
        Random randomizer = new Random();

        pret = randomizer.nextFloat() * 100;
        denumire = titles[randomizer.nextInt(titles.length)];
        autor = names[randomizer.nextInt(names.length)];
        int nrCopii = randomizer.nextInt(10);
        state = new ArrayList<CarteCopie>(nrCopii);
```

```
        for (int i = 0; i < state.size(); i++) {
            state.set(i, new CarteCopie());
        }
    }

    public float PretTotal() {
        return pret * state.size();
    }

    public float GetIntrebuintare() {
        float result = 0f;

        for (int i = 0; i < state.size(); i++) {
            result += state.get(i).getState();
        }

        if (state.size() != 0) {
            result /= state.size();
        }
        return result;
    }

    public float CompareIntrebuintare(Carte other) {
        return this.GetIntrebuintare() - other.GetIntrebuintare();
    }

    public static float ComparePrice(Carte first, Carte second) {
        return first.PretTotal() - second.PretTotal();
    }
}
```

Helper.java

```
package com.company;

import com.google.gson.Gson;
import java.util.Random;
import java.io.*;

public class Helper {

    public static Random randomizer = new Random();
    public static Gson gson = new Gson();

    public static String InputString() {
        BufferedReader box = new BufferedReader(new InputStreamReader(System.in));

        String str = "";
        try {
            str = box.readLine();
```

```
        } catch (Exception e) {
            System.out.println("Failed row reading at Helper.InputString");
            System.out.println(e);
        }

        return str;
    }

    public static int InputInt() {
        boolean success = false;
        int result = 0;

        do {
            try {
                result = (Integer.valueOf(InputString())).intValue();
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                System.out.print("Dati valoarea inca o data: ");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static float InputFloat() {
        boolean success = false;
        float result = 0;

        do {
            try {
                result = (Float.valueOf(InputString())).floatValue();
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                System.out.print("Dati valoarea inca o data: ");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static int InputIntLimit(int min, int max) {
        int result = min - 1;

        do {
            result = InputInt();
            if (result < min || result > max) {
                println("Outside of limits [" + min + ", " + max + "], try
again");
            }
        } while (result < min || result > max);
```

```
        return result;
    }

    public static int InputIntLimit(int min) {
        return InputIntLimit(min, Integer.MAX_VALUE);
    }

    public static float InputFloatLimit(float min, float max) {
        float result = min - 1f;

        do {
            result = InputFloat();
            if (result < min || result > max) {
                println("Outside of limits [" + min + ", " + max + "], try
again");
            }
        } while (result < min || result > max);

        return result;
    }

    public static boolean InputBoolean() {
        boolean success = false;
        boolean result = false;
        do {
            try {
                result = Boolean.parseBoolean(InputString());
                success = true;
            } catch (Exception e) {
                println("Shit's fucked man");
                println(e.toString());
                println("Try again");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static float InputFloatLimit(float min) {
        return InputFloatLimit(min, Float.MAX_VALUE);
    }

    public static void println(String text) {
        System.out.println(text);
    }

    public static void print(String text) {
        System.out.print(text);
    }

    public static String FileRead(String path) {
        String result = "";
```

```
        try {
            BufferedReader box = new BufferedReader(new FileReader(path));

            String line;
            while ((line = box.readLine()) != null) {
                result += line;
            }
            box.close();
        } catch (Exception e) {
            println("Failed file reading at Helper.FileRead");
            println(e.toString());
        }

        return result;
    }

    public static void FileWrite(String path, String text) {
        try {
            File file = new File(path);
            FileOutputStream fileOutputStream = new FileOutputStream(file);

            if (!file.exists()) {
                file.createNewFile();
            }

            byte b[] = text.getBytes();

            fileOutputStream.write(b);
            fileOutputStream.flush();
            fileOutputStream.close();
        } catch (Exception e) {
            println("Failed file writing at Helper.FileWrite");
            println(e.toString());
        }
    }
}
```

Main.java

```
package com.company;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        Carte defaultCarte = new Carte();
        Carte customCarte = new Carte(150f, "The Catcher in the Rye", "J. D.
Salinger", new ArrayList<CarteCopie>(5));
        for (int i = 0; i < 5; i++) {
            customCarte.addState(new CarteCopie());
```

```
        }

        Carte copyCarte = new Carte(customCarte);

        ArrayList<Carte> biblioteca = new ArrayList<>(5);

        biblioteca.add(defaultCarte);
        biblioteca.add(customCarte);
        biblioteca.add(copyCarte);
        biblioteca.add(new Carte());
        biblioteca.add(new Carte());

        customCarte.setState(1, new CarteCopie(1, false));

        biblioteca.set(3, Helper.gson.fromJson(
                Helper.gson.toJson(biblioteca.get(1)),
                Carte.class
        ));

        System.out.println("Diferenta de pret dintre 0 si 1: " +
Carte.ComparePrice(biblioteca.get(0), biblioteca.get(1)));
        System.out.println("Diferenta de pret dintre 2 si 3: " +
Carte.ComparePrice(biblioteca.get(2), biblioteca.get(3)));
        System.out.println("Diferenta de intrebuintare dintre 0 si 1: " +
biblioteca.get(0).CompareIntrebuintare(biblioteca.get(1)));
        System.out.println("Diferenta de intrebuintare dintre 2 si 3: " +
biblioteca.get(2).CompareIntrebuintare(biblioteca.get(3)));
        System.out.println();


        float totalPrice = 0f;
        for (Carte carte : biblioteca) {
            totalPrice += carte.PretTotal();
        }

        System.out.println("Pret total: " + totalPrice);
        System.out.println();

        Carte mostUsed = null;  //nu folosim new Carte() ca sa nu marim nrCarti cu
inca o unitate
        for (Carte carte : biblioteca) {
            if (mostUsed == null || mostUsed.CompareIntrebuintare(carte) < 0) {
                mostUsed = carte;
            }
        }

        for (int i = 0; i < biblioteca.size(); i++) {
            if (mostUsed == null ||
mostUsed.CompareIntrebuintare(biblioteca.get(i)) < 0) {
                mostUsed = biblioteca.get(i);
            }
        }

        System.out.println("Cea mai intrebuintata carte: ");
```

```
        mostUsed.Print();
        System.out.println();

        System.out.println("Nr de carti create: " + Carte.getNrCarti());

        for (Carte carte : biblioteca) {
            //biblioteca[i].SaveToFile("books/" + biblioteca[i].getDenumire() +
".txt");
            Helper.FileWrite(carte.getDenumire() + ".txt",
Helper.gson.toJson(carte));
        }
    }
}
```

Rezultatele rulării programului

```
Diferenta de pret dintre 0 si 1: -750.0
Diferenta de pret dintre 2 si 3: 0.0
Diferenta de intrebuintare dintre 0 si 1: -5.2
Diferenta de intrebuintare dintre 2 si 3: 1.6000004

Pret total: 2250.0

Cea mai intrebuintata carte:
Pret: 150.0
Denumire: The Catcher in the Rye
Autor: J. D. Salinger
Nr copii: 5
Status:
Carte de starea 7
Carte de starea 9
Carte de starea 3
Carte de starea 5
Carte de starea 10


Nr de carti create: 6
```