Darea de seamă la Java: Lucrare de laborator nr. 1

Tema: Realizarea claselor în Java

Varianta: Nr. 16

Clasa: Carte

Efectuat: Țurcanu Cristian, studentul grupei IA1901

Verificat: Epifanova Irina, lector universitar, magistru în informatică

Formularea problemei:

De creat clasa “Carte” cu câteva câmpuri.

Câmpurile obligatorii:

- cîmp cu numere reale, care păstrează preţul exemplarului unei cărţi.
- 2 şiruri de caractere, care păstrează denumirea cărţii şi numele-prenumele autorului.
- cîmp de numere întregi, care indică numărul de copii ai cărţii.
- мector dinamic, care păstrează informaţia despre cît de tare e uzată cartea(de la 0 la 10)
- o variabilă statică care numara ciţi cărţi sunt.

Cîmpuri opţionale : grosimea cărţii(numărul de pagini), gen, culoarea copertei, limbă ş.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)
- câteva constructori cu parametri
- constructor de copiere

- De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).
- De creat metoda pentru a afişa la ecran toată informaţia despre carte (toate câmpurile clasei).
- De descris metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
- De descris metoda, care completeaza toate campurile obiectului cu valori aleatoare.
- De scris o funcţie care va calcula preţul total al cărţii (luînd în consideraţie preţul unei copii şi numărul de copii).
- De scris o funcţie care va calcula cît e de întrebuinţată cartea (media aritmetică nivelului de uzură al copiilor).
- De scris o funcţie care compară 2 cărţi (una fiind de apel şi alta primită ca parametru) după nivelul de întrebuinţare, adică după media aritmetică dintre coeficienţii de uzură a tuturor copiilor cărţii. Cu cît e mai mare coeficientul, cu atît cartea e mai întrebuinţată.
- De scris o funcţie statică, care primeşte ca parametri două cărţi, şi care calculează cu cît prima carte e mai scumpă decît a doua(se ia în consideraţie numărul de copii).

În funcţia main() să se creeze cărţi, folosind toţi constructorii descrişi. După crearea fiecărui obiect al clasei „Carte”, de a afişa la ecran cimpurile lui folosind metodă clasei. Să se creeze un vector dinamic cu cărţi - "Biblioteca". Vectorul să fie iniţializat pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia

despre fiecare carte din bilbiotecă. Să se compare cîteva perechi de cărţi după întrebuinţare şi după preţ. Să se calculeze şi să se afişeze costul total tuturor cărţilor create. Să se găsească cea mai întrebuinţată carte, şi să se afişeze denumirea "campioanei" Ultimul rînd afişat să conţină numărul cărţilor create, folosind variabila statică a clasei.

Pentru nota 10. Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.
- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.
- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “URRRRRRRRRAAAAAAAAAA ! ! ! !” şi de prezentat primul laborator profesorului. :)

Codul sursă

Helper:

```
import java.io.*;

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

        String str = "";
        try {
            str = box.readLine();
        } catch (Exception e) {
            System.out.println("Shit's fucked man");
            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;
    }
}
```

Carte.java

```
import java.util.Random;
import java.io.*;

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

    private static int nrCarti = 0;
    private int state[];

    public Carte() {
        pret = -1f;
        denumire = "---";
        autor = "---";
        nrCopii = 0;

        nrCarti++;
        state = new int[nrCopii];
    }

    public Carte(float pret, String denumire, String autor, int nrCopii, int
state[]) {
        this.pret = pret;
        this.denumire = denumire;
        this.autor = autor;
```

```
            nrCarti++;

            if (nrCopii == state.length) {
                this.nrCopii = nrCopii;
                this.state = state;
            } else {
                System.out.println("WARNING: nrCopii != state.length, reseting to 0");
                this.nrCopii = 0;
                this.state = new int[this.nrCopii];
            }
        }

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

        public Carte(String path) {
            try {
                BufferedReader box = new BufferedReader(new FileReader(path));

                pret = (Float.valueOf(box.readLine())).floatValue();
                denumire = box.readLine();
                autor = box.readLine();
                nrCopii = (Integer.valueOf(box.readLine())).intValue();
                state = new int[nrCopii];

                for (int i = 0; i < nrCopii; i++) {
                    state[i] = (Integer.valueOf(box.readLine())).intValue();
                }
                nrCarti++;

                box.close();
            } catch (Exception e) {
                System.out.println("Shit's fucked man");
                System.out.println(e);
            }
        }

        // TODO: deserialize

        public String Serialize() {
            String result = "";
            result += pret + "\n";
            result += denumire + "\n";
            result += autor + "\n";
            result += nrCopii + "\n";
            for (int i = 0; i < nrCopii; i++) {
                result += state[i] + "\n";
            }
```

```
        return result;
    }

    public void Deserialize(String params) {
        String[] parts = params.split("\n");

        setPret((Float.valueOf(parts[0])).floatValue());
        setDenumire(parts[1]);
        setAutor(parts[2]);
        setNrCopii((Integer.valueOf(parts[3])).intValue());

        for (int i = 0; i < nrCopii; i++) {
            setState(i, (Integer.valueOf(parts[4 + i])).intValue());
        }

        nrCarti++;
    }

    public void SaveToFile(String path) {
        try {
            File file = new File(path);
            FileOutputStream fileOutputStream = new FileOutputStream(file);

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

            byte b[] = Serialize().getBytes();

            fileOutputStream.write(b);
            fileOutputStream.flush();
            fileOutputStream.close();
        } catch (Exception e) {
            System.out.println("Shit's fucked man");
            System.out.println(e);
        }
    }

    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 nrCopii;
    }

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

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

        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 int getState(int id) {
        if (id >= 0 && id < state.length) {
            return state[id];
        }

        System.out.println("Shit's fucked man");
```

```
            System.out.println("Requested invalid id at ***.getState");
            return -1;
        }

        public void setState(int id, int state) {
            if (id >= 0 && id < this.state.length) {
                if (state > 10 || state < 0) {
                    System.out.println("Set invalid state at ***.setState, value must
    be in [0,10]");
                    throw new RuntimeException();
                }
                this.state[id] = state;
            } else {
                System.out.println("Set invalid id at ***.setState");
                throw new RuntimeException();
            }
        }

        public int[] getStateArray() {
            return state.clone();
        }

        public void setStateArray(int[] state) {
            nrCopii = state.length;
            this.state = 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: " + nrCopii);
            System.out.print("Status: ");

            for (int i = 0; i < state.length; i++) {
                System.out.print(state[i] + " ");
            }

            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.length; i++) {
            System.out.println("Dati starea cartii " + i + ": ");

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

        }
    }

    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)];
        nrCopii = randomizer.nextInt(10);
        state = new int[nrCopii];
```

```
        for (int i = 0; i < state.length; i++) {
            state[i] = randomizer.nextInt(10) + 1;
        }
    }

    public float PretTotal() {
        return pret * nrCopii;
    }

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

        for (int i = 0; i < state.length; i++) {
            result += state[i];
        }

        if (state.length != 0) {
            result /= state.length;
        }
        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();
    }
}
```

Main.java

```
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", 5, new int[] {5,7,3,4,1});
        Carte copyCarte = new Carte(customCarte);

        Carte biblioteca[] = new Carte[] {
            defaultCarte,
            customCarte,
            copyCarte,
            new Carte(),
            new Carte()
        };

        customCarte.setState(1, 0);
        biblioteca[3].Deserialize(biblioteca[1].Serialize());
        biblioteca[4].Randomize();
```

```
        for (int i = 0; i < biblioteca.length; i++) {
            biblioteca[i].Print();
            System.out.println();
        }

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


        float totalPrice = 0f;
        for (int i = 0; i < biblioteca.length; i++) {
            totalPrice += biblioteca[i].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 (int i = 0; i < biblioteca.length; i++) {
            if (mostUsed == null || mostUsed.CompareIntrebuintare(biblioteca[i]) <
0) {
                mostUsed = biblioteca[i];
            }
        }
        System.out.println("Cea mai intrebuintata carte: ");
        mostUsed.Print();
        System.out.println();

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

        for (int i = 0; i < biblioteca.length; i++) {
            biblioteca[i].SaveToFile("books/" + biblioteca[i].getDenumire() +
".txt");
        }
    }
}
```

Rezultatele rulării programului

```
Pret: -1.0
Denumire: ---
Autor: ---
```

```
Nr copii: 0
Status:

Pret: 150.0
Denumire: The Catcher in the Rye
Autor: J. D. Salinger
Nr copii: 5
Status: 5 0 3 4 1

Pret: 150.0
Denumire: The Catcher in the Rye
Autor: J. D. Salinger
Nr copii: 5
Status: 5 7 3 4 1

Pret: 150.0
Denumire: The Catcher in the Rye
Autor: J. D. Salinger
Nr copii: 5
Status: 5 0 3 4 1

Pret: 13.8362465
Denumire: wives of the mountain
Autor: Vic Money
Nr copii: 3
Status: 7 6 8

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: -2.6
Diferenta de intrebuintare dintre 2 si 3: 1.4000001

Pret total: 2291.5088

Cea mai intrebuintata carte:
Pret: 13.8362465
Denumire: wives of the mountain
Autor: Vic Money
Nr copii: 3
Status: 7 6 8

Nr de carti create: 6
```