Darea de seamă la Java: Lucrare de laborator nr. 2

Tema: Mostenire in java. Clase abstracte si Interfete

Varianta: Nr. 16

Efectuat: Țurcanu Cristian, studentul grupei IA1901

Verificat: Epifanova Irina, lector universitar, magistru în informatică

Formularea problemei:

1. De creat proiect in Eclipse cu 2 clase: clasa din prima lucrare de laborator si clasa proprie de exceptii. (Cocteil – CocteilException, Sultan – SultanException e t.c.)
2. De adaugat in prima clasa 3 tipuri de prelucrare exceptiilor: standarta, metoda lenesa si try in try
3. Toata prelucrare trebuie sa fie descrisa in metodele clasei proprii de exceptii
4. Toate datele citite de la tastatura de prelucrat in do – while
5. De prelucrat toate greseli posibile la introducerea datelor

Codul sursă

Carte.java

```java



public class Carte {
    public float pret = -1f;
    public String denumire = "---";
    public String autor = "---";
    public int nrCopii = 0;

    public static int nrCarti = 0;
    public int state[] = new int[nrCopii];

    public Carte() {}

    public Carte(float pret, String denumire, String autor, int nrCopii, int
state[]) {
        try {
            setPret(pret);
            setDenumire(denumire);
            setAutor(autor);
            nrCarti++;

            if (nrCopii == state.length) {
                setStateArray(state);
            } else {
                throw new CarteException(nrCopii, state.length);
            }
        } catch (CarteException e) {
            e.Handle();
        }
    }

    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 float getPret() {
        return pret;
    }

    public void setPret(float pret) throws CarteException {
        if (pret < 0f) {
            throw new CarteException(pret);
        }
        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) {

        try {
            if (nrCopii < 0) {
                throw new CarteException(nrCopii);
            }

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

            int minCount = nrCopii < this.nrCopii ? nrCopii : this.nrCopii;

            int temp[] = state.clone();

            state = new int[nrCopii];
            this.nrCopii = nrCopii;

            for (int i = 0; i < minCount; i++) {
                setState(i, temp[i]);
            }
        } catch (CarteException e) {
            e.Handle();
        }
    }

    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) {
        try {
            try {
                if (id >= 0 && id < this.state.length) {
                    if (state > 10 || state < 0) {
                        throw new CarteException(state, true);
                    }
                    this.state[id] = state;
                } else {
                    System.out.println("Set invalid id at ***.setState");
                    throw new ArrayIndexOutOfBoundsException();
                }
            } catch (ArrayIndexOutOfBoundsException e) {
                throw new CarteException(e);
            }
        } catch (CarteException e) {
            e.Handle();
        }
    }

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

    public void setStateArray(int[] state) {
        nrCopii = state.length;

        for (int i = 0; i < state.length; i++) {
            setState(i, state[i]);
        }
        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: ");
        try {
            setPret(Helper.InputFloatLimit(0f));
        } catch (CarteException e) {
            e.Handle();
        }

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

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

        System.out.print("Dati nr. copii: ");
        setNrCopii(Helper.InputIntLimit(0));

        for (int i = 0; i < state.length; i++) {
            System.out.println("Dati starea cartii " + i + ": ");
            setState(i, Helper.InputIntLimit(0, 10));
        }
    }

    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" };

        pret = Helper.randomizer.nextFloat() * 100;
        denumire = titles[Helper.randomizer.nextInt(titles.length)];
        autor = names[Helper.randomizer.nextInt(names.length)];
        nrCopii = Helper.randomizer.nextInt(10);
        state = new int[nrCopii];

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

CarteException.java

```
   public class CarteException extends Exception {
       private float wrongPret = 0f;
       private int wrongNrCopii = 3;
       private int wrongState = 5;
       private int wrongStateArrayLength = -1;
       private Object standardException;
```

```
    public CarteException(float wrongPret) {
        this.wrongPret = wrongPret;
    }

    public CarteException(int wrongNrCopii) {
        this.wrongNrCopii = wrongNrCopii;
    }

    public CarteException(int wrongState, boolean unusedParam) {
        //unusedParam is ofc just to get a different param signature
        this.wrongState = wrongState;
    }

    public CarteException(int wrongNrCopii, int wrongStateArrayLength) {
        this.wrongNrCopii = wrongNrCopii;
        this.wrongStateArrayLength = wrongStateArrayLength;
    }

    public CarteException(Object standardException) {
        this.standardException = standardException;
    }

    public void Handle() {
        if (wrongPret < 0f) {
            Helper.println("pret should be >= 0f");
        }

        if (wrongNrCopii < 0 && wrongStateArrayLength == -1) {
            Helper.println("nrCopii should be >= 0");
        }

        if (wrongState > 10 || wrongState < 0) {
            Helper.println("state should be between 0 and 10 inclusively");
        }

        if (wrongNrCopii != wrongStateArrayLength && wrongStateArrayLength != -1)
{
            Helper.println("nrCopii and state.length should be the same");
        }

        if (standardException instanceof ArrayIndexOutOfBoundsException) {
            Helper.println("accessed index out of bounds of state");
            Helper.println(standardException.toString());
        }
    }

}
```

Helper.java

```
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 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

```
public class Main {

    public static void main(String[] args) {
        //Carte carte = Helper.gson.fromJson(Helper.FileRead("carte.json"),
Carte.class);

        Carte carte = new Carte();

        carte.Edit();

        try {
            carte.setPret(-1f);
        } catch (CarteException e) {
            e.Handle();
        }
        carte.setNrCopii(2);

        carte.setState(3, 5);

        carte.Print();

        Helper.FileWrite("carte.json", Helper.gson.toJson(carte));
    }
}
```

Rezultatele rulării programului

```
Dati pretul: -1
Outside of limits [0.0, 3.4028235E38], try again
5
Dati denumirea: wasd
Dati autorul: ijkl
Dati nr. copii: -2
Outside of limits [0, 2147483647], try again
2
Dati starea cartii 0:
-5
Outside of limits [0, 10], try again
1
Dati starea cartii 1:
7
pret should be >= 0f
WARNING: nrcopii == this.nrCopii, no changes
Set invalid id at ***.setState
accessed index out of bounds of state
java.lang.ArrayIndexOutOfBoundsException
Pret: 5.0
Denumire: wasd
Autor: ijkl
Nr copii: 2
Status: 1 7
```