Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra Informatică Aplicată

**RAPORT**

Lucrarea de laborator nr.5

Tehnici Avansate de Programare

A efectuat:

st. gr. C-171 D. Melniciuc

A verificat:

dr., conf.univ. M. Oșovschi

Chişinău 2019

**Crearea excepţiilor**

Fiecare caz excepţional este în conformitate cu anumite clase. În cazul în care clasa corespunzătoare nu există, atunci ea poate fi creată. Excepţii sunt moştenite de la clasa de bază **Throwable**  şi clasele sale derivate **Error** şi **Exception** din pachetul **java.lang.**

În Java atunci când o excepţie este generată întotdeauna este creat un obiect de tip corespunzător, care trebuie să fie prelucrat de către un anumit bloc de cod. Atunci când se creează o metodă prelucrarea excepţiei poate fi inclusă în codul, care poate genera această metodă.

Există trei moduri de a crea excepţiile :

- blocuri **try – catch ;**

- declaraţia excepţiei în secţiunea **throws**  a metodei şi transmiterea apelantului

- capturarea excepţiei, convertirea la altă clasă şi repetarea apelului

Examinare primului mod are loc în următorul exemplu. La I/O se foloseşte o clasa IOException.

### Оperatorul throw

Operatorul **throw** se utilizeauă pentru generarea excepţiilor. În acest caz se utiliyeayă obiectul clasei **Throwable sau obiectul clasei derivate**, sau referinţile lor. Sintacsa instrucţiunii **throw**:

**throw Obiect*Throwable;***

După realizarea acestui operator, realiuarea codului programului se întrerupe. Cel mai apropiat bloc **try** găseşte realizarorul de excepţie necesar **catch**. Dacă este găsit , începe realiuarea lui, altfel este controlat următorul bloc înclus **try**.

În exemplul următor se crează un obiect excepţie şi apoi operatorul **throw** generează excepţia prelucrată în **catch** în cere la rîndul său este generată o altă excepţie.

* 1. **Exceptiile proprii**

Avem posibilitatea de a crea proprii excepţii ca o clasă derivată a clasei Exception şi apoi să le folosim în prelucrarea situaţiilor care apar la activarea metodei. De exemplu:

/\* exemplu # 6 : metodă, care generează o excepţie creată de programator: \*/

**public** **class** MyBasic {

**static** **int** result(**int** i)

**throws** MyResultException {

**try** {

i = 1 / i;

**throw** **new** MyResultException("Excepţia proprie");

}

**catch** (ArithmeticException e) {

**throw** **new** MyResultException(

"MyResultException bazată pe o altă excepţie ", e);

}}

**public** **static** **void** main(String[] args) {

**try** {

**int** a = result(8);

}

**catch** (MyResultException e) { System.out.println(e.toString());}}}

Atunci când este imposibil să se calculeze valoarea necesară se generează obiestul **ArithmeticException,** realizatorul căruia la rândul său generează exscepţia **MyResultException,** folosită ca o excepţie proprie, care are două argumente. Primul - un mesaj care poate fi afişat în fluxul de eroare, şi al doilea - o excepţie, care aduce la apelul excepţiei proprii. Acest cod arată cum se poate salva informaţii în o excepţiile proprii. Avantajul în acest caz este că dacă metoda activată va dori să ştie motivul real de apel pentru **MyResultException,** ea doar trebuie să apeleze metoda **getHiddenException().** Acesta permite metodei apelate să deschidă sau să lucreze cu o excepţie specifică sau este destul de prelucrat **MyResultException**  .

***Var 5:*** definiţi clasa Stack. Întroduceţi o secvenţă de caractere şi afisaţ-o în ordine inversă

***Codul:***

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

import java.io.\*;

import java.lang.RuntimeException;

import java.util.Timer;

import java.util.TimerTask;

class text extends JFrame implements ActionListener {

static JFrame f;

static JButton b;

static JLabel l;

static JTextArea jt;

text() {}

public static void cls() {

try {

new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();

} catch (Exception e1) {

System.out.println(e1);

}

}

public static void main(String[] args) throws InterruptedException {

System.out.println("wait 3 seconds");

Thread.sleep(3000);

cls();

System.out.println("programul se ruleaza");

f = new JFrame("");

l = new JLabel("nothing entered");

l.setBounds(50,50,100,30);

b = new JButton("submit");

text te = new text();

b.addActionListener(te);

jt = new JTextArea(10, 10);

jt.setBounds(100,400,100,30);

JPanel p = new JPanel();

p.add(jt);

p.add(b);

l.setText("introduceti o secventa");

p.add(l);

f.add(p);

f.setSize(300, 300);

f.show();

}

public void actionPerformed(ActionEvent e) {

String s = e.getActionCommand();

if (s.equals("submit")) {

String st = jt.getText();

String rev = "";

for (int i = st.length() - 1; i >= 0 ; i--) {

rev = rev + st.charAt(i);

}

System.out.println("\nRaw: " + st);

System.out.println("Reversed: " + rev);

l.setText(rev);

boolean b = true;

try {

for (int i = st.length() - 1; i >= 0 ; i--) {

if (st.charAt(i) == ' ')

throw new IOException();

}

} catch (IOException e2) {

l.setText("introduceti doar o secventa");

b = false;

}

try {

File file = new File("in.txt");

FileWriter fw = new FileWriter(file);

PrintWriter pw = new PrintWriter(fw);

pw.println("Raw: " + st);

pw.println("Reversed: " + rev);

pw.close();

} catch (IOException exept) {

exept.printStackTrace();

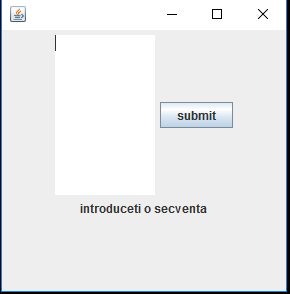
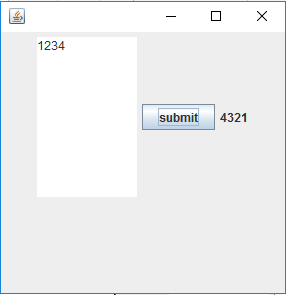
}

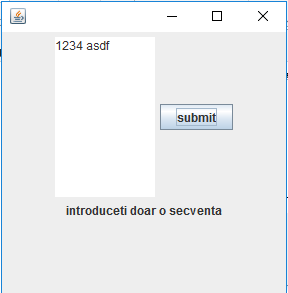
}

}

}

***ScreenShot:***



***Concluzie:***

*Dupa efectuarea laborotorului am dobintid experienta in limbajul de programere java lucrind cu exceptii, utilizind exceptiile standarte din java si crearea propriilor exceptii.*