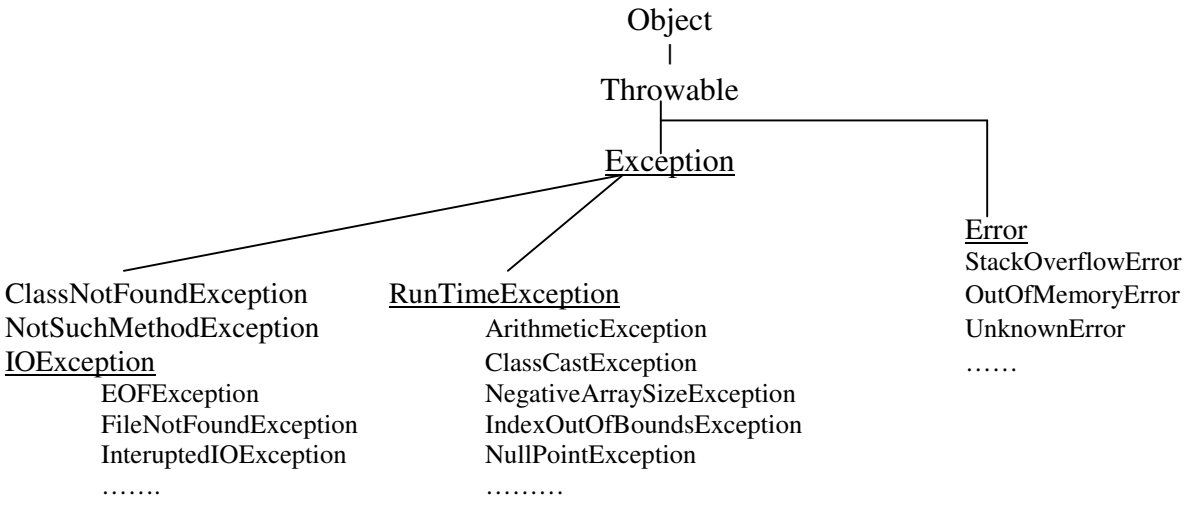
Situacijų valdymas

Situacija – tai įvykis, nutraukiantis įprastą programos darbą. Tai gali būti klaida programoje arba specialiai programuotojo numatyta situacija.

Kiekvienai situacijai Java'oje sukuriamas atitinkamos klasės objektas. Pagrindinės situacijų klasės yra:



Exception klasės situacijas programuotojas gali "sugauti" ir apdoroti. Gali sukurti ir savvo situacijų klases. **RunTimeException** klasės situacijų yra labai daug ir jos gali įvykti programos vykdymo metu. Programuotojas jas gali apdoroti, bet neprivalo. Likusios **Exception** klasės situacijos privalo arba būti apdorotos, arba prsiųstos apdorojimui kitur. Tai sutikrina kompiliatorius.

Error klasės situacijų programuotojas valdyti negali, kadangi tai dažniausiai susiję su JVM klaidomis (tolimesni JVM veiksmai neprognozuojami).

Įvykus situacijai, pvz., dalybai iš nulio, ją visada pagaus "bevardis" situacijų apdorotojas. Jis paprasčiausiai atspausdins kokia situacija ir kokioje programoje įvyko ir programa baigs darbą. Tai nėra patogu. Norint nukreipti tolimesnius įvykius sava linkme, t.y., sugauti situaciją ir ją pačiam apdoroti, arba tiesiog "kultūringai" užbaigti programą, javoje naudojama :

- **try** – **catch** blokai;
- sakinys **throw**;
- žodelis **throws**;
- **finally** blokas.

P V Z :

```
/** Skaityti tekstinį failą po eilutę (iki failo galo) ir išvedinėti į ekraną.
    Failo vardas imamas iš komandinės eilutės.
    Situacijos :
     1. Iššaukti paaiškinančią situaciją, kai parametrų sąrašas tuščias;
     2. "Sugauti" situaciją, kai nurodyto failo nėra;
     3. Įvykus kuriai nors kitai situacijai atlikti jos "trasavimą".
     4. Kosmetika (finally blokas).
**/
import java.io.*;
public class SkaitoFaila {
   public static void main(String args[]) throws IOException {
      // throws IOException reikia metodui close(), esančiame finally
      // (arba kitaip finally turi tureti try-catch bloka)
      BufferedReader skaityti = null;
```

```
        try {
            if (args.length == 0) {
                throw (new Exception("Įveskite : SkaitoFaila failoVardas"));
                // programa įvykdys catch (Exception e) bloką ir baigs darbą.

            }
            skaityti = new BufferedReader(new FileReader(args[0]));
            String eilute = new String();
            // skaito po eilutę iki failo galo ir išvedinėja :
            while ((eilute = skaityti.readLine()) != null) {
                System.out.println(eilute);
            }
        }
        catch (FileNotFoundException e) {
            System.out.println("Duomenu failas " + args[0] + " nerastas.");
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
            e.printStackTrace();
        }
        finally {
            if (skaityti != null) {
                System.out.println("Duomenu failas uzdarytas");
                skaityti.close();
            } else {
                System.out.println("Failas nebuvo atidarytas");
            }
        }
    }
}
```

1. **try – catch** blokai

Programos kodas, kuriame gali įvykti situacija, patalpinamas į bloką **try**, po kurio talpinamas blokas **catch**, nusakantis kurią situaciją pagauti ir kokius tuo atveju atlikti veiksmus :

```
try {
        // veiksmai
}
catch (GaudomaSituacija ob) {
        // veiksmai jai įvykus
}
```

Jei **try** bloke tikimasi kelių situacijų, tai po jo dedami keli **catch** blokai. Svarbu išlaikyti hierarchiją – subklasės **catch** eina prieš superklasės **catch**. Taigi Veiksmai-2 niekada nebus atlikti :

```
try {
        // veiksmai
} catch (Exception e) {
        // veiksmai-1
} catch (ArithmeticException e) {
        // veiksmai-2
}
```

Pastabėlės:

1. **Try** blokai gali turėti vidinius **try** blokus su savo **catch**.
2. Suradus ir įvykdžius **catch** bloką, likusieji **catch** blokai ignoruojami.

3. Su **catch**(Exception e) galima sugauti bet kurią situaciją. Informaciją apie tai, kokia tai buvo situacija galima gauti su šios klasės metodu *getMessage*(), o visą ivykių medį – metodu *printStackTrace*().

2. Sakinys **throw**

Leidžia iššaukti situaciją rankiniu būdu. Sintaksė :

trow *ThrowableArbaJosSubklasiųObjektas*;

Šis objektas gali būti sukurtas kaip naujas su funkcija new prieš tai, arba tiesiogiai **catch** antraštėje. Po sakinio **throw** tolimesnė veiksmų seka nutraukiama ir tikrinamas artimiausias **catch**. Jei situacija ten ne ta, tai tikrinamas sekantis **catch** blokas ir t.t. Jei nerasta – suveiks bevardis apdorotojas.

3. Žodelis **throws**

Jei metode gali susidaryti situacija, bet metodas pats jos neapdoroja, tai metodo antraštėje po žodelio **throws** būtina išvardinti šias situacijas. Tokiu būdu jų apdorojimas perduodamas (“nusviedžiamas”) šį metodą iššaukusiai klasei.

Pvz :

```
public void aMetodas () throws FileNotFoundException {
   // …
   // Jei šiame metode įvyktų situacija “failas nerastas”,
   // tai ji bus apdorota jį iššaukusio metodo cMetodas catch bloke
}
public void bMetodas () throws EOFException {
   // …
   // Jei šiame metode įvyktų situacija “rastas failo galas”,
   // tai ji nebus apdorota jį iššaukusiame metode cMetodas, nes jis neturi tokio catch bloko.
   // Ši situacija bus permesta main metodui.
}

public void cMetodas () throws EOFException {
   // ……
   try {
      aMetodas();
      bMetodas();
} catch (FileNotFoundException c) {
      // ….
   }
   // ….
}
public static void main(String args[]) {
// ……
   try {
      cMetodas();
   } catch (IOException e) {
      // Jei kviečiant metodą cMetodas metode bMetodas įvyktų situacija “rastas failo galas”,
      // tai ji patektų į šį bloką, nes EOFException klasė yra IOException klasės vaikas.
      // ….
   }
}
```

4. Blokas **finally**

Įvykus situacijai visgi daugeliu atveju normalus programos ciklas bus pažeistas. Gali likti, kad kai kas labai svarbaus nepadaryta, pvz, neuždaryri skaitymo/rašymo failai. Čia gelbsti blokas **finally**. Jis vykdomas visada – įvyko situacija ar ne, buvo ji sugauta ir apdorota ar ne.

P.S.

1. Po **try** bloko vietoje **catch** gali eiti **finally** blokas. Jei nėra nė vieno, bus klaida.
2. **Finally** vykdomas prieš išeinant iš metodo, kuriame jis yra paskelbtas.

Nuosavos situacijos

Jei programuotojo netenkina esamos javos situacijos, jis gali jas papildyti savomis. Dažniausiai nuosavos situacijos kuriamos kaip klasės, paveldinčios **Exception** klasę. Tokios klasės dažniausiai turi du konstruktorius (be ir su String parametru), bet gali tureti ir pagalbinių metodų.