ИЗКЛЮЧЕНИЯ В JAVA

гл.ас. д-р Мария Евтимова

https://github.com/marias83837/JavaPresentations

Обработка на грешки чрез изключения

Изключение:

1. Обект, който дефинира необичайна или предизвикваща грешки ситуация
2. Обектно- ориентирано решение за обработка на грешки
3. Изключението се хвърля от програмата или от системата при изпълнение
4. Механизъм, даващ информация за грешки по време на изпълнение
5. Може да бъде хванато и обработено
6. Концепция за хващане и обработка на грешки в коректния контекст

Грешка

1. Подобна на изключение, но обикновено представя непоправима ситуация
2. Не може да се хване

1. Хвърляне на изключение

При възникване на изключителна ситуация методът информира извикващия го метод чрез:

throw<израз>

<израз>- от клас Throwable

2.Хващане на изключение

try

<блок try>

catch(<декларация на обект на изключение>)

<блок catch>

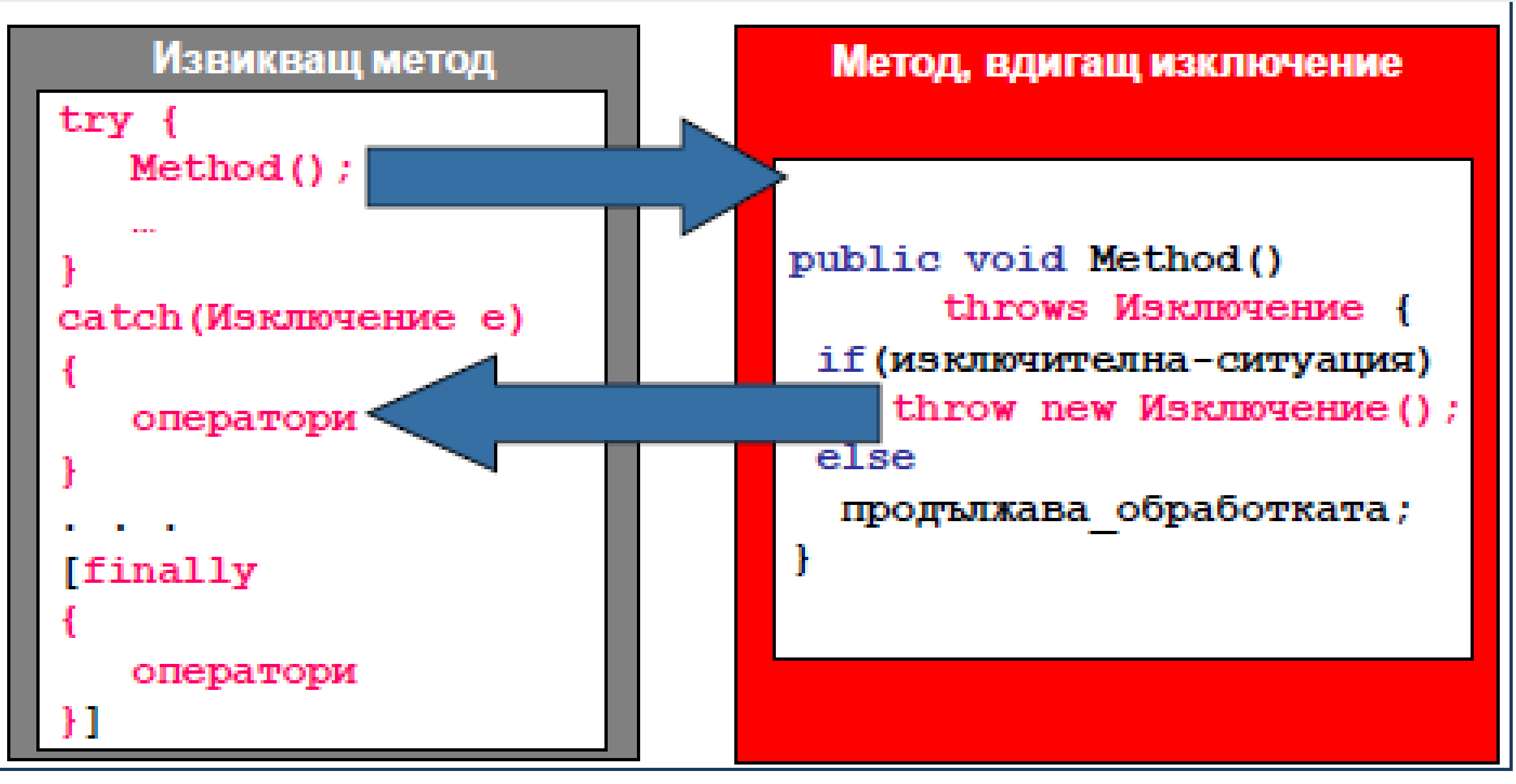
.....

finally

< блок finally>

<декларация на обект на изключение>- от клас Throwable

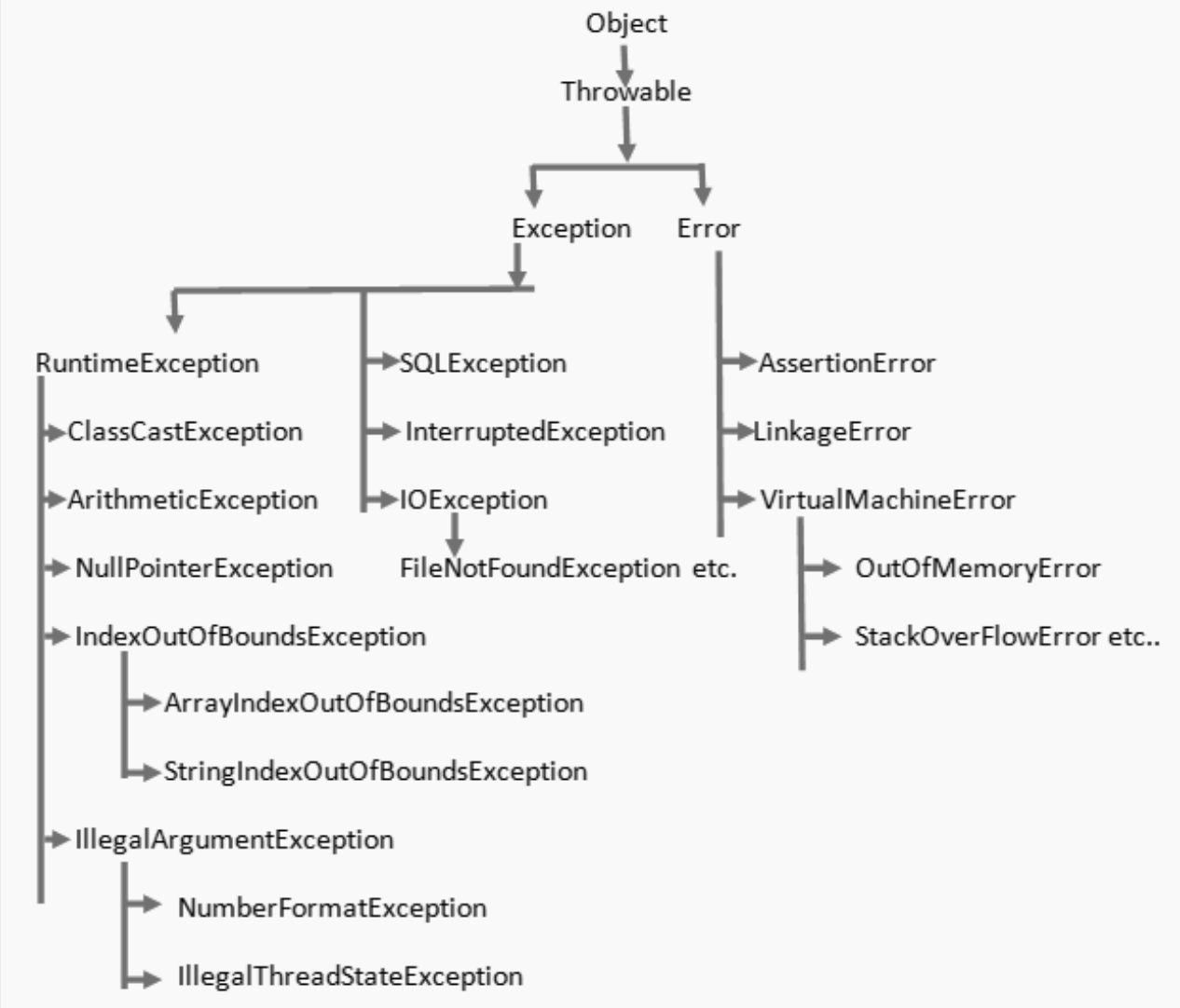
Изпълнява се **<блок try>**. Ако той хване изключение, контролът се предава на подходящата catch клауза и изключението се обработва в **<блок catch>.** Клаузата finally не е задължителна и винаги се изпълнява **<блок finally>**


Извикващ метод
try {
Method();
…
}
catch(Изключение e)
{
оператори
}
. . .
[finally
{
оператори
}]
Метод, вдигащ изключение
public void Method()
throws Изключение {
if(изключителна-ситуация)
throw new Изключение();
else
продължава_обработката;
}

Опростяване и стандартизиране на обработката на грешки с изключения

1. Кодът е по- чист – показва реалната работа, която трябва да се извърши при обработените грешки
2. Кодът не изисква проверка за върнатите стойности от всеки извикан метод и транслиране на кодовете за грешки- използва външен блок, който хваща изключенията от всички извикани методи
3. Блокът finally позволява освобождаване на ресурсите дори при възникване на изключение

Йерархия на класовете за изключения



Клас java.lang.Exception- сигнализира, че може да се предизвика изключение

- **public Exception(String s)**

Конструира Exception със съобщение s

- **public String getMessage()**

Връща съобщение за грешка на обекта от тип **Throwable**

- **public final Class getClass()**

Връща класа на обекта по време на изпълнение

Клас java.ioi.IOException

Сигнализира, че може да възникне входно/изходно изключение;

то трябва да се обработи от оператор **try** или трябва да се включи в списъка на клаузата **throws** на метод, който може да го разпространи

Обработка на изключения

- Изключението не се обработва
- Изключението се обработва в мястото на възникване
- Изключението се обработва на друго място в програмата

Пример: Дефинира собствено изключение OutOfRangeException (дадена стойност е извън зададена област)

При наследяване на класа Exception

```
public class OutOfRangeException extends Exception{
public OutOfRangeException(String message){
//message е съобщение, което определя изключение
super(message);
}
}
```

Хвърляне на изключение - ArrayIndexOutOfBoundsException

```
public class Example{
   public static void main(String args[]) {
      try {
         int massive[] = new int[5];
         System.out.println("Достъп до елемент
шест :" + massive[6]);
      } catch (ArrayIndexOutOfBoundsException e)
{
         System.out.println("Хвърлено
изключение:" + e);
      }
      System.out.println("Извън проверката");
   }
}
```

Прехвърляне на изключение

Даден метод може да хване изключение, да го обработи според контекста му и да го прехвърли обратно в стека чрез ключовата дума throw изключение, така че по- нататък отново може да стане грешка.

-InputMismatchException (следващия низ не може да преобразува в **int** или е извън областта)

-NoSuchElementException

- IllegalStateException

Методът main() на тестващия клас Example обработва изключението OutOfRangeException чрез try-catch блок

```
import  java.util.Scanner;
class OutOfRangeException extends
Exception{
    void checkValue(int value,int
MIN,int MAX)
        throws OutOfRangeException{
            if(value<MIN||value>MAX){
                throw new
OutOfRangeException();
            }
    }
```

Методът main() на тестващия клас Example обработва изключението. Проверява y дали е в интервал [-10;10]

```
public class Example{

    public static void main(String[] args) {

        OutOfRangeException x = new
OutOfRangeException();

        int MIN = -10;

        int MAX = 10;

        int value=0;

        try{

            Scanner scan = new Scanner(System.in);

            System.out.println("Enter y: ");

            value=scan.nextInt();

            x.checkValue(value,MIN,MAX);

        }catch(OutOfRangeException e){

            System.out.println(e+" y не е въведен в граници от -
10 до 10");

        }catch(Exception e){

            System.out.println(e+"сгрешен е вида на въведения
тип");

        }finally{

            System.out.println("y:"+value+"");

        }

    }

}
```

Хващане на много изключения

Базовият клас трябва да се обработи последен- ако неговият блок **catch** е първи, останалите блокове **catch** ще бъдат недостижими и останалите изключения няма да се обработят

```
public class Example{
public static void main (String[] args){
int [] A= new int[] {1,2,3,4,5,6,7,8,9,10};
int i=11, k=3;
try{
int temp= A[i]/k;
A[i+1]= Integer.parseInt("10"+temp);
}catch (ArrayIndexOutOfBoundsException e){

System.out.println("Грешка при индексиране");
}catch(ArithmeticException e){
System.out.println("Грешка при деление на нула");
}catch(NumberFormatException e){
System.out.println("Грешка при преобразуване на низ в цяло число");
}catch(Exception e){
System.out.println("Грешка"+e);
}
}
}
```

Сравняване на техниките за обработка на грешки

- **Връщане код на грешка- стандартен начин за управление на грешките**

Недостатък- няма гаранция, че извикващият метод ще провери върнатия код на грешка

- **Предимства на обработка на изключения пред връщане на код**

1. Гаранция за обработка на изключението- управлението се предава обратно в стека и извикващият метод е принуден да го обработи
2. Управлението на грешките в коректния контекст- осигурява разширяемост- най- значимото предимство
3. Подобряване четимостта на кода