ИЗКЛЮЧЕНИЯ

ЛЕКЦИОНЕН КУРС "ПРОГРАМИРАНЕ НА JAVA"

Distributed @learning center

СТРУКТУРА НА ЛЕКЦИЯТА

- Стандартна обработка на грешка
- Класове на грешки
- try-catch-оператор
- finally блок
- Дефинирано от потребителя създаване на изключения

ИЗКЛЮЧЕНИЯ

- Особени условия по време на изпълнение на програмите
- Безсмислено продължаване на изпълнението:
  предимно run-time грешки → прекъсване на програмата

Примери:

- Array: грешен Index a[i]
- Вход: очаква се число
  - въвежда се буква
- Достъп до обект (извикване на методи):
  - съществува само 'null'-Object
- Липсва клас
  (след компилация: класът изтрит)

Цел: Въпреки изключението трябва да е възможна по-нататъшна работа

ПРИМЕРИ: RUN-TIME ГРЕШКИ ПРЕДИЗВИКВАТ ИЗКЛЮЧЕНИЯ

1 Какви резултати?

```
class Excep   {
   public static void main (String[] args) {
        int i = Integer.parseInt(args[0]);
        System.out.println("i = " + i);
   }
}
```

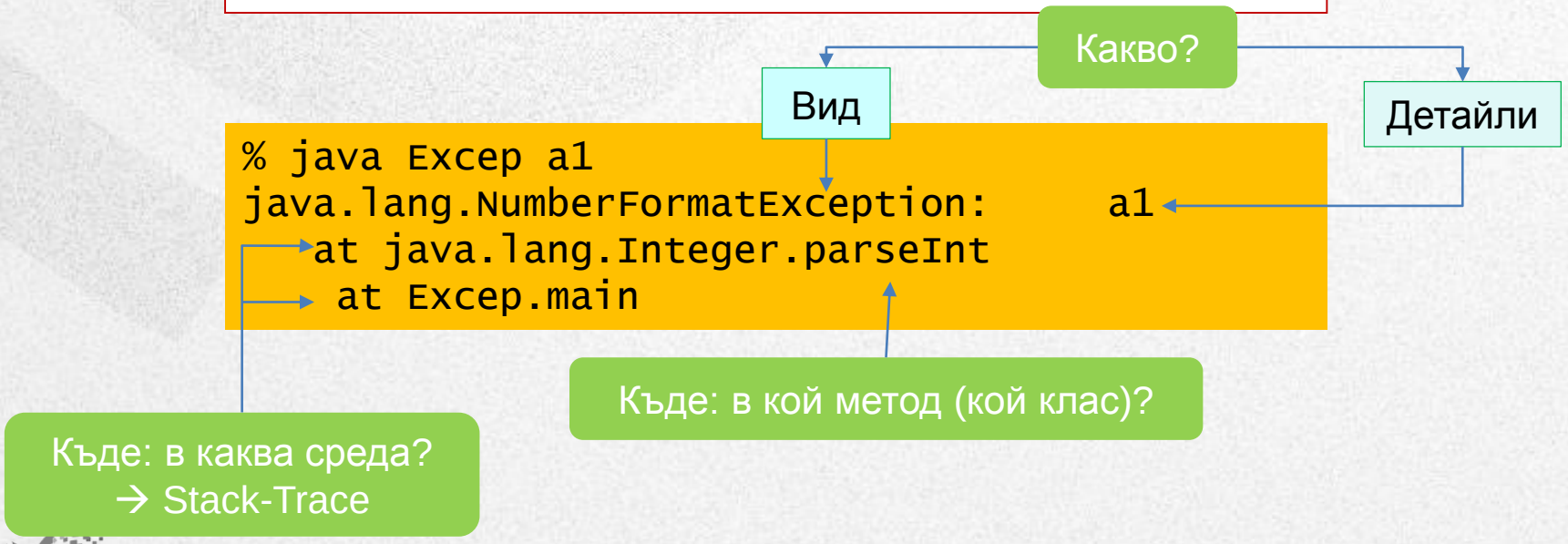
```
% java Excep 3
i = 3

% java Excep
java.lang.ArrayIndexOutOfBoundsException: 0
   at Excep.main(Compiled Code)

% java Excep a1
java.lang.NumberFormatException: a1
   at java.lang.Integer.parseInt
    at Excep.main
```

СЪОБЩЕНИЯ ЗА ИЗКЛЮЧЕНИЯ

```
class Excep   {
   public static void main (String[] args){
        int i = Integer.parseInt(args[0]);
        System.out.println("i = " + i);
   }
}
```



STACK-TRACE

Обратен списък на всички извикани, но още не завършени методи

```
class Excep
    static int makeIntFromString (String s){
         return Integer.parseInt(s);
    }
    public static void main (String[] args){
         int i = makeIntFromString(args[0]);
         System.out.println("i = " + i);
    }
}
```

```
% java Excep a1
java.lang.NumberFormatException:      a1
    at java.lang.Integer.parseInt
    at Excep.makeIntFromString
    at Excep.main
```

Извикване 3

Извикване 2

Извикване 1

Посока на изхода (Stack)

СЪОБЩЕНИЯТА НЕ СА ЕДИННИ

```
% java Excep a1
java.lang.NumberFormatException:     a1
at java.lang.Throwable
at java.lang.Exception
at java.lang.RuntimeException
at java.lang.IllegalArgumentException
at java.lang.NumberFormatException
at java.lang.Integer.parseInt
at Excep.makeIntFromString
at Excep.main
```

Конструктори за създаване на обекти за изключения
→ йерархия на класове (run-time-API-Stack-Trace се извиква от Java-Interpreter)

Stack-Trace (приложна програма)

ОБРАБОТКА НА ИЗКЛЮЧЕНИЯ В JAVA: ОБЕКТНО-ОРИЕНТИРАНО РЕШЕНИЕ

```
class Excep   {
   static int makeIntFromString (String s)   {
         return Integer.parseInt(s);
   }
   public static void main (String[] args)   {
         int i = makeIntFromString(args[0]);
         System.out.println("i = " + i);
   }
}
```

Ред на извикване (с актуални параметри):

```
% java Excep a1  4  61
```

main({"a1", "4", "61"})

makeIntFromString("a1")

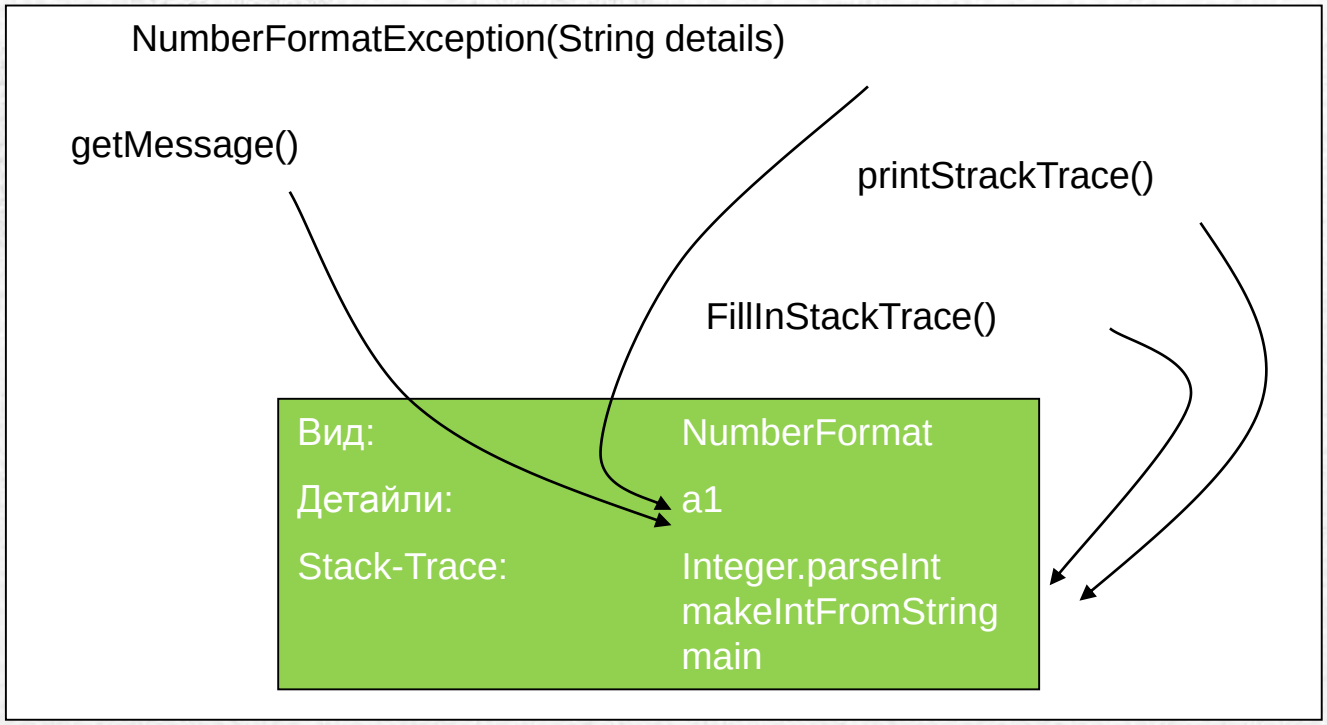
Integer.parseInt("a1")

Java-VM-Mashine:
- открива изключителни ситуации
- Създава обект
  (един клас на изключения)

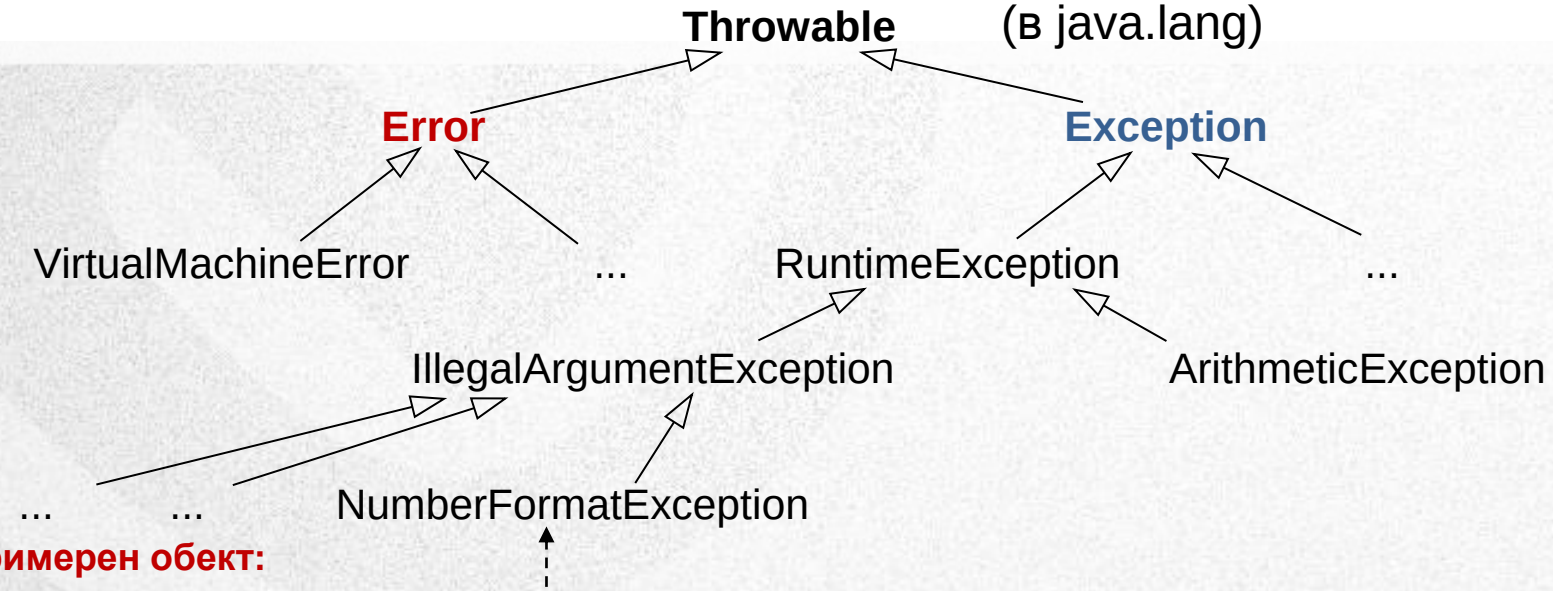
ОБЕКТИ ЗА ИЗКЛЮЧЕНИЯ

Java-VM-Mashine:
създава обект (един клас на изключения)

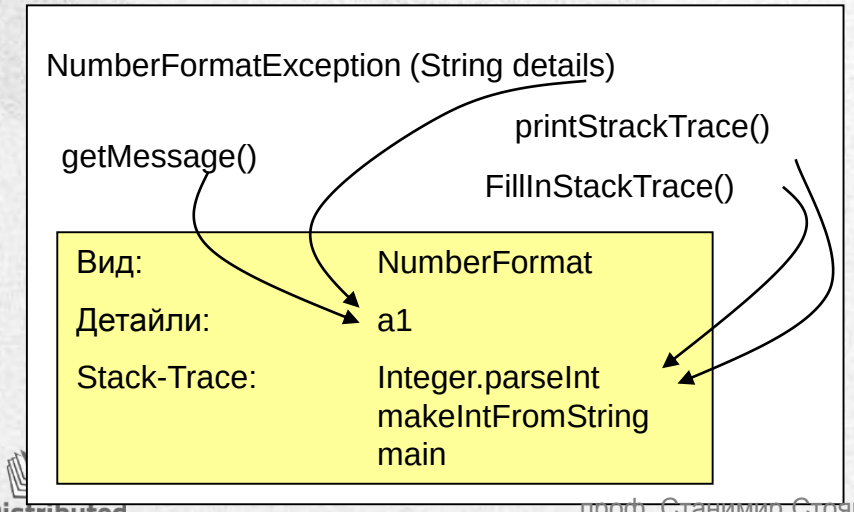
Обект от класа NumberFormatException:



КЛАСОВЕ НА ИЗКЛЮЧЕНИЯ: ЙЕРАРХИЯ



Примерен обект:



за вид по изключение:
специален клас

Throwable:	общ клас за изключения
Error:	по-тежки грешки (обикновено неотстраними)
Exception:	обработваеми

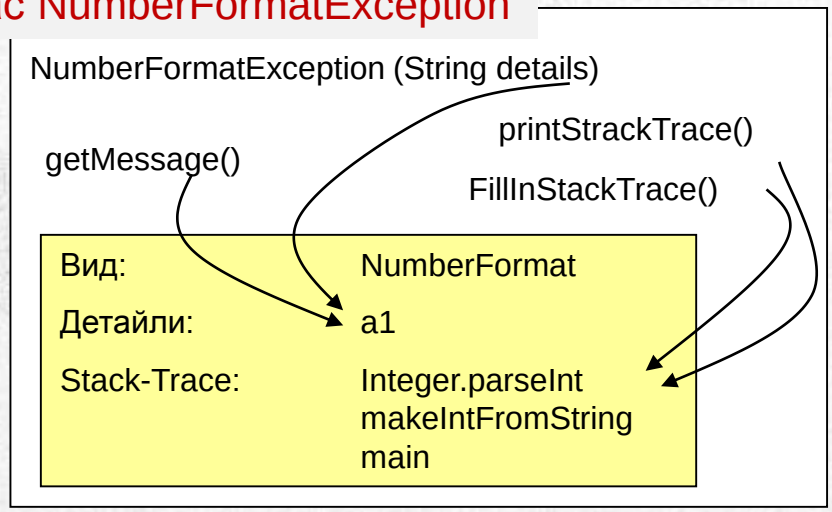
ОБРАБОТКА НА ИЗКЛЮЧЕНИЯ

1. Какво значи обработка на изключения?

Оценка на създадените обекти на изключения

2. Кой оценява информацията?

Обект от клас NumberFormatException



- Java-Standard-обработване на грешки (JVM):
  - Изход на информацията, съхранена в обекта на изключения (виж горе)
  - Прекъсване на програмата
- Дефинирано от потребителя обработване на грешки:

try-catch - оператор

TRY-CATCH: ПРИМЕР

Критични оператори
→възможна изключителна ситуация

```
class TryCatch    {
   public static void main (String[] args)    {
        try    {
            int i = Integer.parseInt(args[0]);
            System.out.println("i = " + i);
        } catch (NumberFormatException e)    {
            System.out.println("As argument ...");
        }
   }
}
```

Обработка

TRY-CATCH: ЕФЕКТ В ДЕТАЙЛИ

```
class TryCatch    {
   public static void main (String[] args)    {
        try    {
          int i = Integer.parseInt(args[0]);
          System.out.println("i = " + i);
        }
```

→ създава се (ев.) обект на изключение o от тип (клас) t :

Ако t отговаря на типа на параметъра на catch:

- обект o се свързва с e
- catch–оператор се изпълнява

```
        catch (NumberFormatException e)    {
          System.out.println("As argument ...");
        }
   }
```

Програмистът определя, какво става след изключенията (тук: NumberFormatException) и програмата се продължава регулярно !

TRY-CATCH: ИЗВИКВАНЕ

```
class TryCatch   {
   public static void main (String[] args)   {
        try   {
           int i = Integer.parseInt(args[0]);
           System.out.println("i = " + i);
        } catch (NumberFormatException e)   {
            System.out.println("As argument an
                  int-value needed");
        }
   }
}
```

```
% java TryCatch a1
```
грешен аргумент

```
As argument an int-value needed
% java TryCatch
```
липсващ аргумент

```
java.lang.ArrayIndexOutOfBoundsException: 0
 ...
```

ОБРАБОТКА НА ПОВЕЧЕ ИЗКЛЮЧЕНИЯ

```
class TryCatchAll   {
   public static void main (String[] args)   {
        try   {
           int i = Integer.parseInt(args[0]);
           System.out.println("i = " + i);
        }
        catch (NumberFormatException e)   {
           System.out.println("As argument ...");
        }
        catch (ArrayIndexOutOfBoundsException e)   {
           System.out.println("Call with a Par.");
        }
        catch (Throwable e)   {
           e.printStackTrace();
        }
   }
}
```

НОРМАЛНО ПРОДЪЛЖАВАНЕ НА РАБОТАТА

1 Без try-catch? програмата завършва след настъпване на изключение (нерегулярно)

```
class TryCatchAll   {
   public static void main (String[] args)   {
        try   {  ...
        }
       catch (NumberFormatException e)   {...
        }
       catch (ArrayIndexOutOfBoundsException e)   {...
        }
       catch (Throwable e)   {  ...
        }
        System.out.println ("Program ends normally ");
   }
}
```

Съобщението се извежда винаги

FINALLY-БЛОК

1 Какъв резултат?

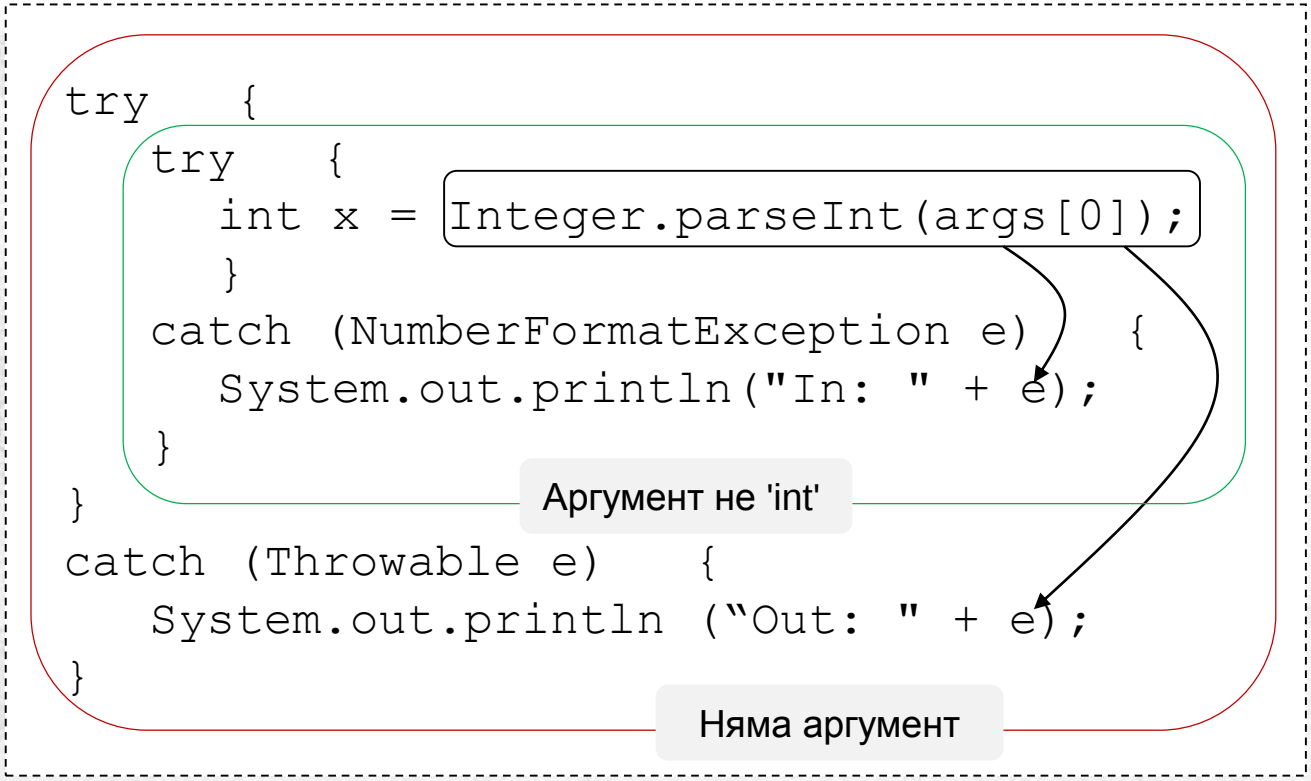
finally-Code се обработва **винаги** (с/без изключение)

Идея: общ код за заключителни действия

```
class Finally   {
   public static void main (String[] args)   {
        int i = 1000, j = 0;
        try   {  i /= j;
        } catch (ArithmeticException e)   {
           System.out.println(e);
        } finally  {
           System.out.println (i + " / 0 undef.");
        }
   }
}
```

```
% java Finally
. . .
1000 / 0 undef.
```

ВГРАДЕНИ TRY-ОПЕРАТОРИ



Ако изключението във вътрешния catch не се появява:

търси заобикалящия try-catch

(съотв. try-catch на един извикан метод)

ПРИМЕР: СИГУРНО ЧЕТЕНЕ НА ЧИСЛА

```
public static int intRead() {
  int number = 0; boolean ok = false;
  System.out.print(„Number input: ");
  while (!ok)   {
     try   {
        ok = true;
        number = Keyboard.readInt();
     }
     catch ( NumberFormatException e )   {
        System.out.print(„No number!! repeat:");
        ok = false;
     }
  }
  return number;
}
```

ВЪПРОСИ

1. Кой създава обектите на изключенията?
2. Кой ги оценява?

```
class Excep   {
   public static void main (String[] args)   {
        int i = Integer.parseInt(args[0]);
        System.out.println("i = " + i);
   }
}
```

```
% java Excep 3
i = 3

% java Execp
java.lang.ArrayIndexOutOfBoundsException: 0
   at Ausnahme.main(Compiled Code)

% java Excep a1
java.lang.NumberFormatException: a1
   at java.lang.Integer.parseInt
    at Excep.main
```

КОЙ СЪЗДАВА ИЗКЛЮЧЕНИЯТА ?

- VJM (Java-Interpreter):

  напр. - ненамерен клас
  - грешен Array-Index

- Методи на Java-API

  напр. Integer.parseInt

```
public static int parseInt (String s)
      throws NumberFormatException
```

- Потребителска програма

ИЗКЛЮЧЕНИЯ: СЪЗДАВАНЕ + ОБРАБОТКА

ДЕФИНИРАНИ ОТ ПОТРЕБИТЕЛЯ ИЗКЛЮЧЕНИЯ

```
class Weekend extends Exception   {
   Weekend(String text)   {
      super(text);   }
}
```

ex = new Weekend ("Sunday")
ex.getMessage() → "Sunday"

Java-API

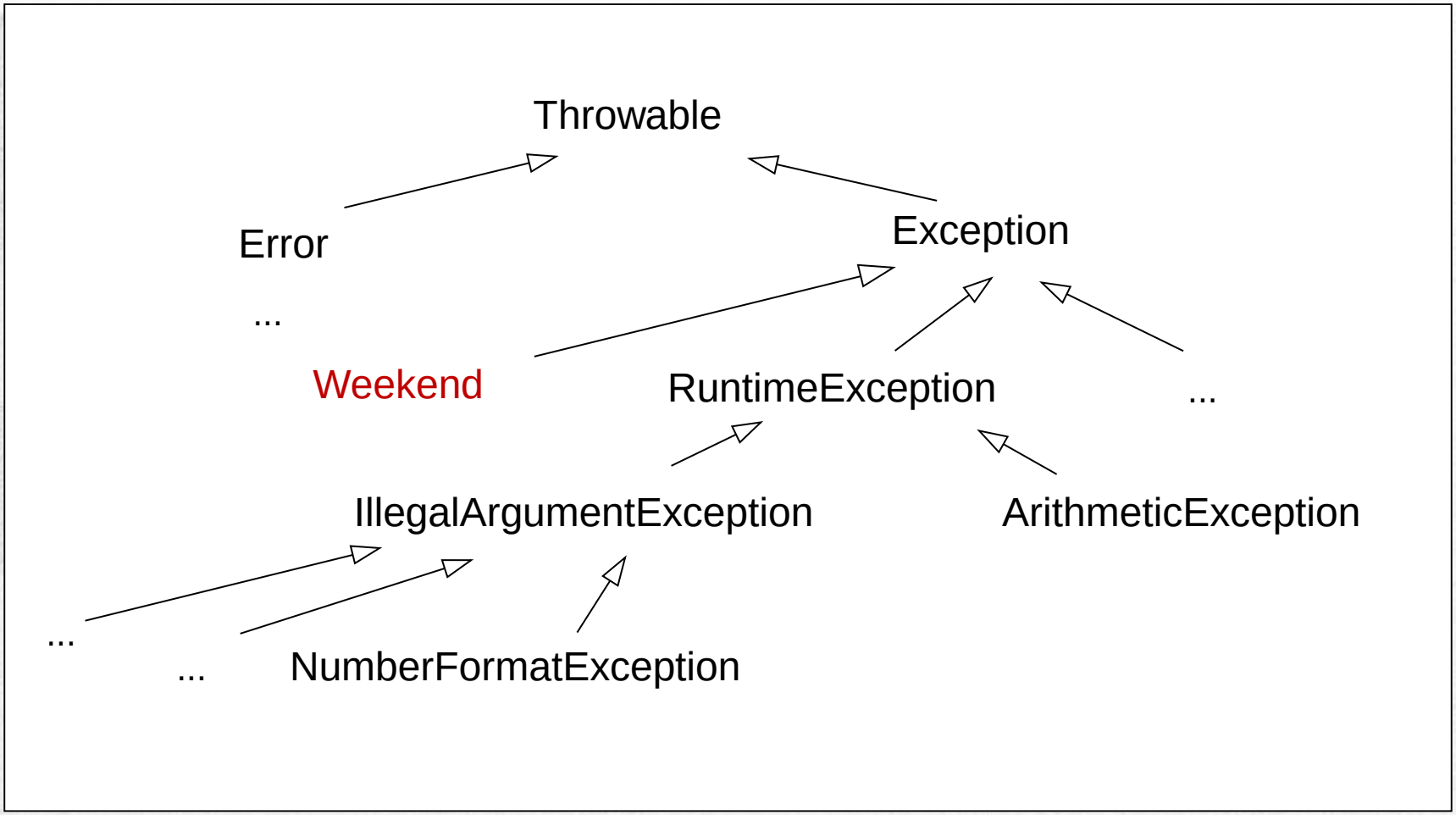
```
class Exception   {

   public Exception (String s)
      // creates message with content s

   public String getMessage ()
      // returns Info.s
   ...
}
```

В края на седмицата програмата работи по различен начин

НОВ КЛАС НА ИЗКЛЮЧЕНИЕ: 'WEEKEND'



ИЗПОЛЗВАНЕ: ДЕФИНИРАНО ОТ ПОТРЕБИТЕЛЯ ИЗКЛЮЧЕНИЕ

```
public static void main (String[] args)   {
   try   {
      todayOpen();
   } catch (Weekend ex)    {
      System.out.println ( ex.getMessage()+ "s closed.");
   }

}
```

Нормален случай

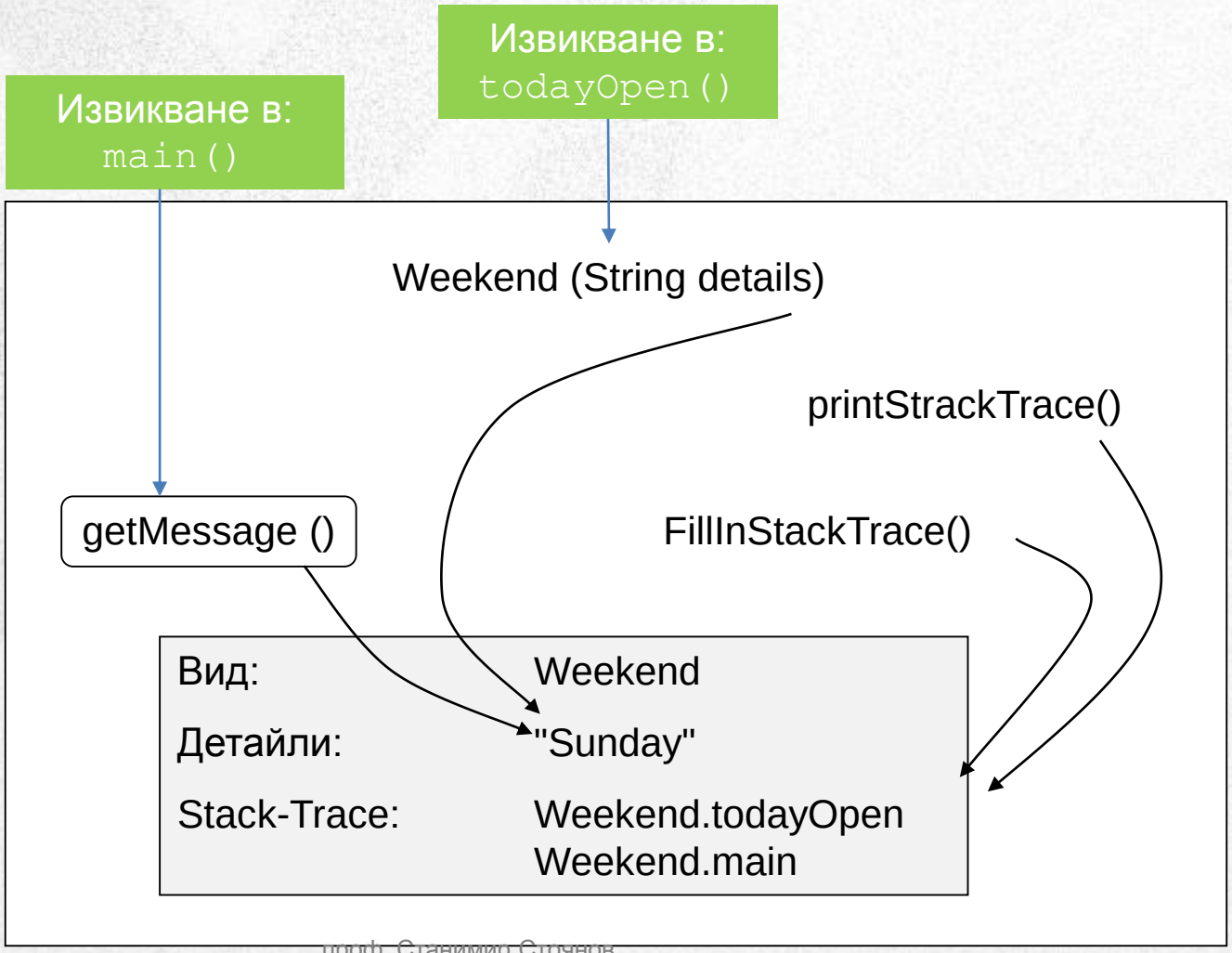
Изкл. ситуация, напр. "Sunday"

```
static void todayOpen() throws Weekend {
   int day = Calendar ... // API-Class
   if (day == Calendar.SUNDAY)
      throw new Weekend("Sunday");
   if (day == Calendar.SATURDAY)
      throw new Weekend("Saturday");
   System.out.println(„Today open.");
}
```

Методът специфицира изключение

Методът създава изключение

ОБЕКТ НА ИЗКЛЮЧЕНИЕ СЛЕД THROW NEW WEEKEND ("SUNDAY")



API-CLASS CALENDAR: ДОСТАВЯ АКТУАЛНИЯ ДЕН ОТ СЕДМИЦАТА

```
static void todayOpen () throws Weekend {

   int day = Calendar.getInstance().get(Calendar.DAY_OF_WEEK);

   if (day == Calendar.SUNDAY)
        throw new Weekend("Sunday");

   if (day == Calendar.SATURDAY)
        throw new Weekday("Saturday");

   System.out.println(„Today opened.");
}
```

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “ИЗКЛЮЧЕНИЯ”

Distributed
@learning center