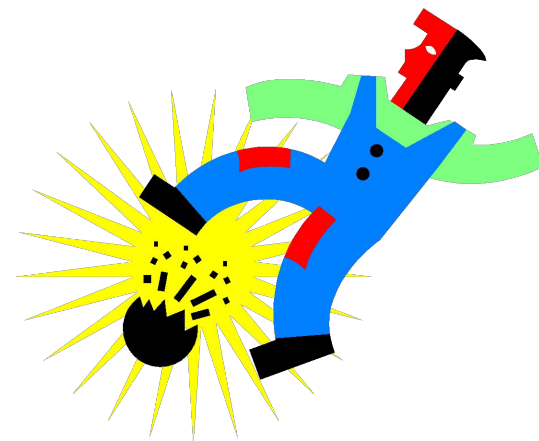




# Занятие 5

---



## Исключения



# Цели

---

- *Дать определение исключения*
- *Описать обработку исключений*
- *Описать блоки `try`, `catch` и `finally`*
- *Объяснить использование нескольких блоков `catch`*
- *Использовать вложенные блоки `try` / `catch`*
- *Объяснить использование ключевых слов `throw` и `throws`*
- *Создавать исключения, определённые пользователем*
- *Изучить использование инструкции `Assert`*

# Что такое исключение?

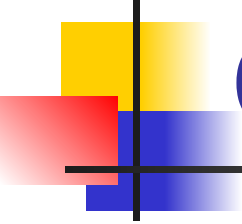
- Если во время выполнения программы возникает ошибка, то это называют исключительной ситуацией или просто исключением.

пример:

**4 / 0 =**



- При возникновении исключения программа немедленно завершается, и управление возвращается операционной системе.
- Обработка исключений выполняется с целью идентификации ошибок и их перехвата.



# Обработка исключений – Общий пример

---

- Фрагмент, записанный на псевдокоде:

```
.....  
IF B IS ZERO GO TO ERROR  
C = A/B  
PRINT C  
GO TO EXIT
```

```
ERROR:  
DISPLAY "DIVISION BY ZERO"
```

```
EXIT:  
END
```

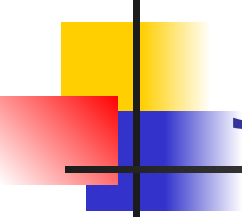
Блок, который  
обрабатывает ошибку



# Обработка исключений в языке Java

---

- Обработка ошибок в Java выполняется на основе модели обработки исключений.
- Появление ошибки вызывает исключение, которое перехватывается в специальном блоке кода.
- Обработку исключений в Java обеспечивают классы `Error` и `Exception`.
- Когда возникает исключение, создается экземпляр исключения, который передается методу, получающему и обрабатывающему информацию об исключении.
- Исключения класса `Error` являются внутренними исключениями
  - Напр. – чтение файла с дискеты, которая не вставлена в дисковод

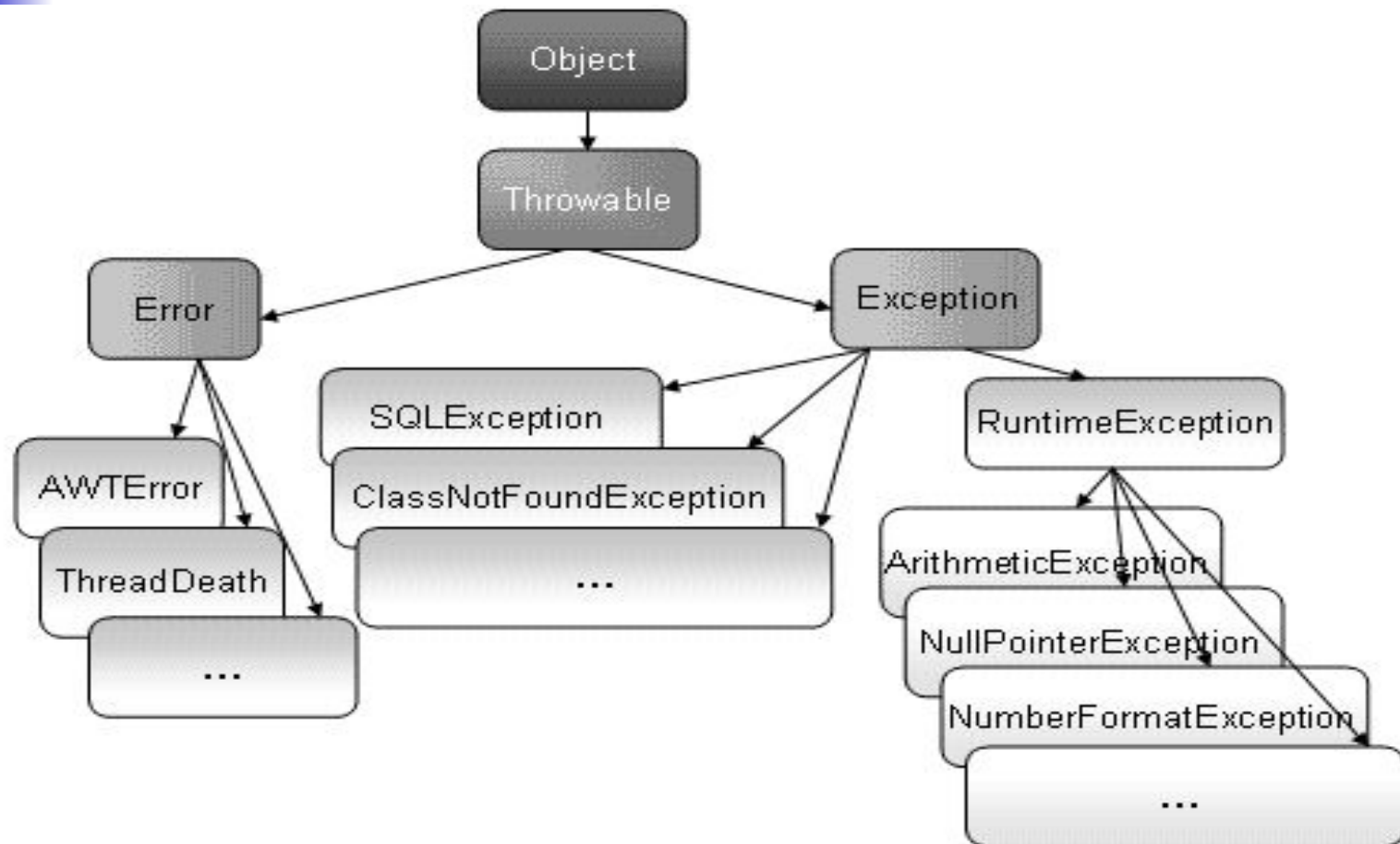


# Обработка исключений в языке Java (продолжение)

---

- В Java обработка исключений управляется посредством пяти ключевых слов: `try`, `catch`, `throw`, `throws` и `finally`.
- Инструкции программы, в которых должны отслеживаться исключения, помещаются в блок `try`.
- Используя ключевое слово `catch`, программист может перехватить исключение и обработать его некоторым разумным образом.
- Для генерации исключения «вручную» мы используем ключевое слово `throw`.
- Вариант `throws` используется в методе для указания на то, что этот метод будет генерировать исключения.
- В блоке `finally` мы можем определить код, который безусловно необходимо выполнить перед возвращением из метода.

# Иерархия классов исключений





# Модель обработки исключений

---

- Обработка с помощью пяти ключевых слов – `try`, `catch`, `throw`, `throws` и `finally`
- Два способа обработки исключений в языке Java:
  - Инструкции, которые могут генерировать исключения, размещаются в блоке `try`, а инструкции обработки исключений располагаются в блоке `catch`
  - Метод может быть определён таким образом, что при возникновении исключительных ситуаций все эти исключения игнорируются



# Блоки try и catch – Пример

```
class ExceptionDemo
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        try
```

```
        {
```

```
        }
```

```
    } catch
```

```
    {
```

```
        arguments given
```

```
    }
```

```
}
```

**ВЫВОД**

C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac ExceptionDemo.java

E:\Java\JavaExamples>java ExceptionDemo

No arguments given!

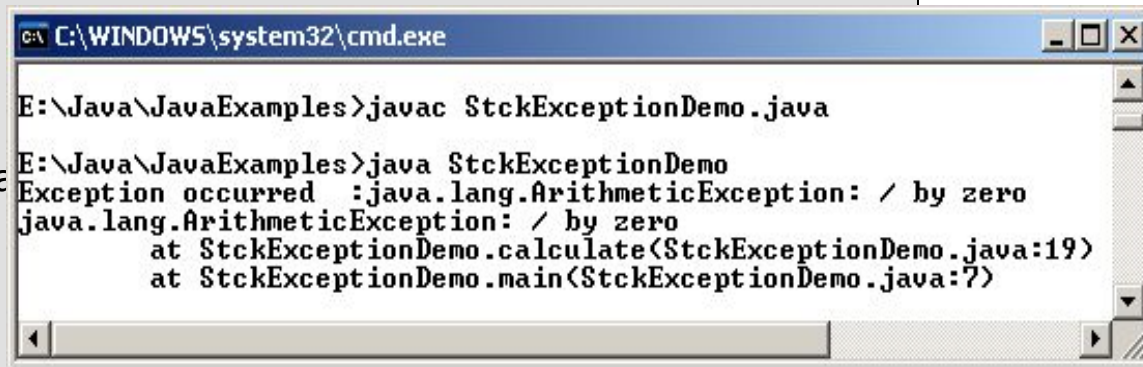
# Пример программы, демонстрирующей обработку исключений

```
public class StckExceptionDemo
{
    public static void main(String args[])
    {
        try
```

**Вывод**

```
        e.toString());
    }
}
```

```
static int calculate( int no, int no1)
{
    int num = no / no1;
    return num;
}
```



```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac StckExceptionDemo.java

E:\Java\JavaExamples>java StckExceptionDemo
Exception occurred :java.lang.ArithmeticException: / by zero
java.lang.ArithmeticException: / by zero
    at StckExceptionDemo.calculate(StckExceptionDemo.java:19)
    at StckExceptionDemo.main(StckExceptionDemo.java:?)
```



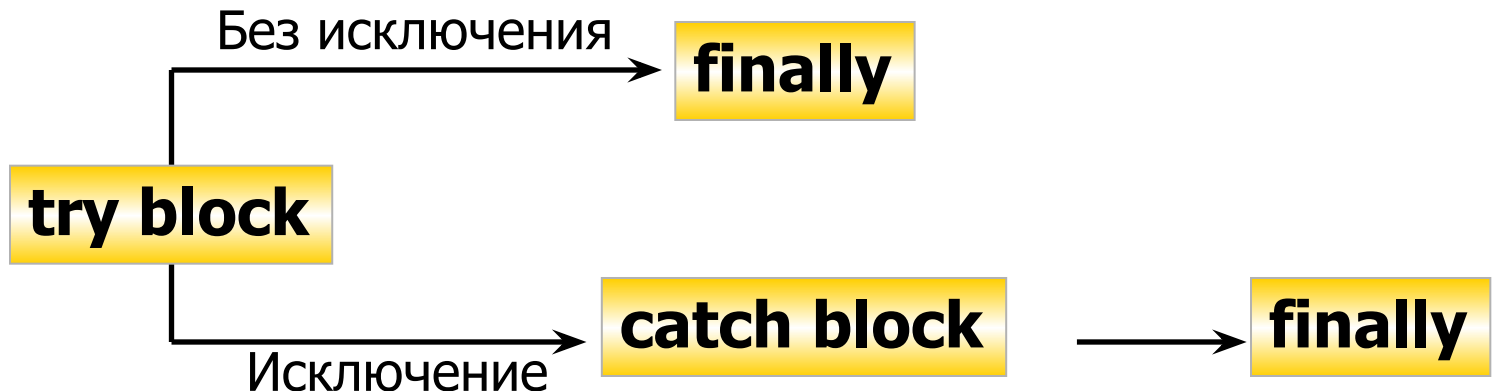
# Методы, используемые в примере

---

- **toString()** извлекает строковое (`String`) представление информации, хранящееся в объекте `Exception`
- **printStackTrace()** используется для вывода причины исключения, а также строки кода, которая сгенерировала это исключение

# Блок `finally`

- Гарантирует, что при возникновении исключения вся работа по очистке памяти будет выполнена
- Используется в сочетании с блоком `try`
- Гарантируется выполнение независимо от того, возникло исключение или нет
- При генерации исключения блок `finally` отработает даже в том случае, если отсутствует соответствующий блок `catch`

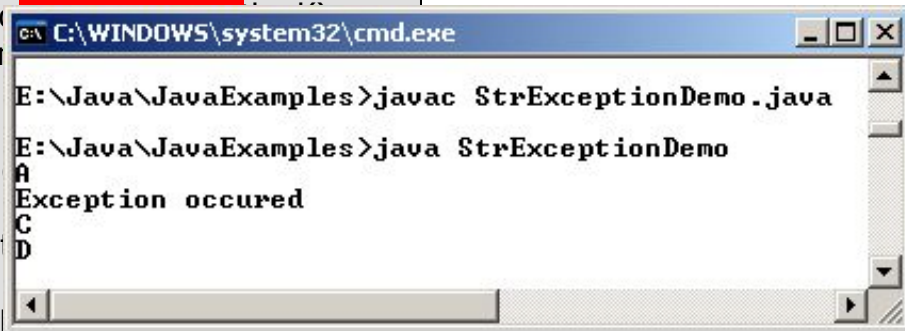


# Блок finally – Пример

```
class StrExceptionDemo
{
    static String str;
    public static void main(String s[])
    {
        try
        {
            System.out.println("A");
            StrExceptionDemo.throwException();
            System.out.println("B");
        }
        catch (NullPointerException e)
        {
            System.out.println("C");
        }
        finally
        {
            System.out.println("D");
        }
    }

    static void staticLengthmethod()
    {
        System.out.println(str.length());
    }
}
```

ВЫВОД



```
C:\WINDOWS\system32\cmd.exe
E:\Java\JavaExamples>javac StrExceptionDemo.java
E:\Java\JavaExamples>java StrExceptionDemo
A
B
C
D
```



# Несколько блоков catch

---

- Один фрагмент кода может генерировать несколько ошибок.
- Поэтому возможно наличие нескольких блоков catch.
- Порядок выполнения инструкций `catch` соответствует порядку их расположения.

```
.....  
try{  
}  
catch(ArrayIndexOutOfBoundsException e) {  
}  
catch(Exception e) {  
}  
.....
```

- `ArrayIndexOutOfBoundsException`, являющийся подклассом класса `Exception`, обрабатывается первым.

# Пример

```
class MultipleCatch
{
    public static void main(String args[])
    {
        try
        {
            String num = args[0];
            numValue);
        }
        catch(ArrayIndexOutOfBoundsException nb)
        {
            System.out.println("Not a number!");
        }
        catch(NumberFormatException nb)
        {
            System.out.println("Not a number!");
        }
    }
}
```

ВЫВОД



```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac MultipleCatch.java
E:\Java\JavaExamples>java MultipleCatch
No arguments given!
E:\Java\JavaExamples>java MultipleCatch a
Not a number!
E:\Java\JavaExamples>java MultipleCatch 6
The square is: 36
```



# Вложенные блоки `try - catch`

---

- Иногда часть одного блока может вызывать ошибку, а весь этот блок может стать причиной другой ошибки.
- В этом случае обработчики исключений должны быть вложенными.
- При использовании вложенных блоков `try` внутренний блок `try` выполняется первым.
- Если отсутствует соответствующий блок `catch`, то проверяются блоки `catch` внешних блоков `try` до тех пор, пока не будут выполнены все вложенные инструкции `try`.



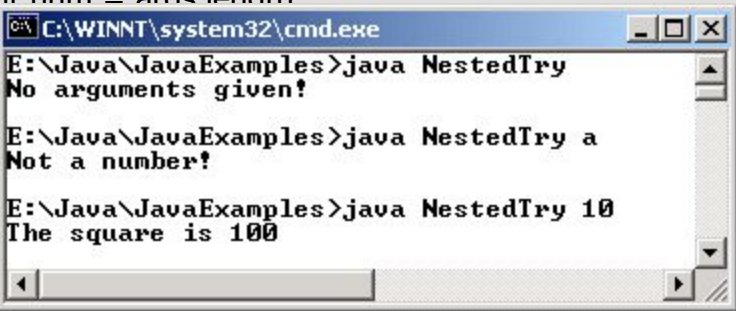
# Пример

```
class NestedTry
{
    public static void main(String args[])
    {
        try
        {
            int num = args.length;
            t
            {
                E:\Java\JavaExamples>java NestedTry
                No arguments given!

                E:\Java\JavaExamples>java NestedTry a
                Not a number!

                E:\Java\JavaExamples>java NestedTry 10
                The square is 100
            }
            System.out.println("Not a number! ");
        }
        catch(ArrayIndexOutOfBoundsException ne)
        {
            System.out.println("No arguments given! ");
        }
    }
}
```

**Вывод**





# Использование throw и throws

---

- Исключения генерируются с использованием ключевого слова `throw`.

```
try
{
    if(flag<0)
    {
        throw new NullPointerException();
    }
}
```

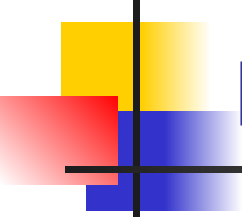
- Один метод может генерировать (throw) более одного исключения.



# Использование throws

```
public class Example
{
    public void exceptionExample()
    {
        try
        {
            // statements
            check();
        }
        catch(Exception e)
        {
            //statements
        }
    }
    // multiple exceptions separated by a comma
    void check() throws NullPointerException,
    NegativeArraySizeException
    {
        if (flag < 0)
            throw new NullPointerException();
        if(arrsize < 0)
            throw new NegativeArraySizeException();
    }
}
```

- Метод `check()` включает ключевое слово `throws`
- Такие методы должны вызываться внутри блоков `try / catch`



# Исключения, определённые пользователем

---

- Встроенные исключения не всегда достаточны для перехвата всех ошибок.
- Поэтому возникает необходимость в классе исключения, определяемого пользователем.
- Он должен быть подклассом класса `Exception`
- Новый тип исключения может быть перехвачен отдельно от прочих подклассов из группы `Throwable`.
- Определённые пользователем классы исключений, которые создаются, наследуют все методы класса `Throwable`.

# Пример

```
class ArraySizeException extends NegativeArraySizeException
{
    void checkSize() throws ArraySizeException
```

```
    {
        if(size < 0)
            throw new ArraySizeException();
    }
}
```

```
class
{
    public
```

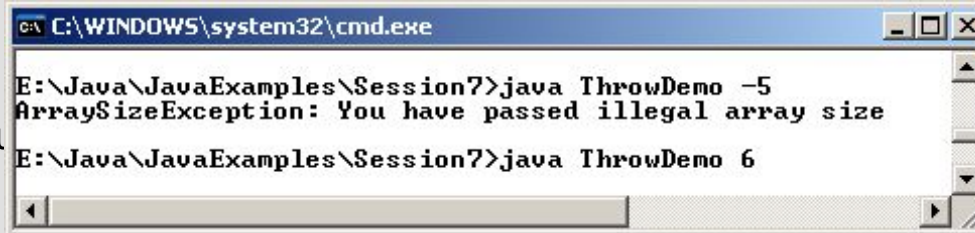
```
    int[] arr = new int[3];
    int count = 0;
    while(count < 3)
    {
        arr[count] = count;
        count++;
    }
}
```

```
    public static void main(String[] args)
    {
        try
        {
            ThrowDemo t = new ThrowDemo();
            t.checkSize();
        }
        catch(ArraySizeException e)
        {
            System.out.println(e);
        }
    }
}
```

```
    UserDefinedException(Integer.parseInt(arg[0]));
}
```

```
    }
    catch(ArraySizeException e)
    {
        System.out.println(e);
    }
}
```

**Вывод**



```
C:\WINDOWS\system32\cmd.exe
E:\Java\JavaExamples\Session7>java ThrowDemo -5
ArraySizeException: You have passed illegal array size
E:\Java\JavaExamples\Session7>java ThrowDemo 6
```

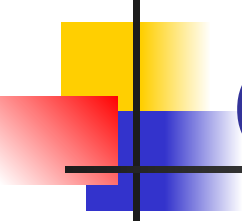


# Программирование с утверждениями assertions

---

- Утверждения (Assertions) позволяют программисту проверять предположения о программе.
- Способы записи инструкций `assertion`:
  - `assert expression;`
  - `assert expression1:expression2;`
- Выражение `expression` в первой форме имеет тип `boolean` и выполняется, если даёт результат `true` (истина), а если выражение ложно (`false`), то генерируется ошибка `AssertionError`.
- Во второй форме выражение `expression1` является логическим (`boolean`), а выражение `expression2` может иметь любое значение за исключением того, что оно не может вызывать метод `void`.
- Если при вычислении `expression1` получается результат `false` (ложь), то генерируется ошибка `AssertionError`. Значение выражения `expression2` будет передано в конструктор `AssertionError`, и это значение будет использовано при выводе сообщения об ошибке.

# Программирование с утверждениями assertions (продолжение)



- Ситуации для использования утверждений assertion:
  - Внутренние инварианты (Internal invariants): утверждение assertion может быть использовано при наличии инварианта
  - Инварианты потока управления (Control Flow Invariants): утверждение assertion может быть помещено в любом месте, в которое не может быть передано управление
  - Предусловия (Preconditions), постусловия (post conditions) и инварианты класса (class invariants)
- Команда для компиляции файлов, которые используют инструкции assertion:
  - `javac -source 1.4 filename`



# Подведение итогов

---

- Если во время выполнения программы возникает ошибка, то это называют исключительной ситуацией или просто исключением.
- Исключение `Exception` возникает во время выполнения последовательности кода.
- Каждое генерируемое исключение обязательно должно быть перехвачено, иначе приложение сразу же завершается.
- Обработка исключений позволяет собрать операции по обработке ошибок в одном месте.
- Java использует блоки `try` и `catch` для обработки исключений.





# Подведение итогов (продолжение)

---

- Инструкции в блоке `try` генерируют исключения, а блок `catch` обрабатывает их.
- Несколько блоков `catch` могут использоваться вместе для раздельной обработки различных типов исключений.
- Ключевое слово `throws` используется для объявления списка исключений, которые метод может сгенерировать.
- Ключевое слово `throw` используется для обозначения факта возникновения исключения.
- Инструкции в блоке `finally` выполняются независимо от того, возникло исключение или нет.