JAVA

ПЛАТФОРМЕНО-НЕЗАВИСИМИ ПРОГРАМНИ ЕЗИЦИ

Лекция 3. Управляващи конструкции

доц. д-р инж. Румен П. Миронов

1. Условни оператори

Условен оператор if

```
if <условен израз> {
        .....
        .....
        .....
}
```

блок от оператори

```
if <условен израз>
        .....
```

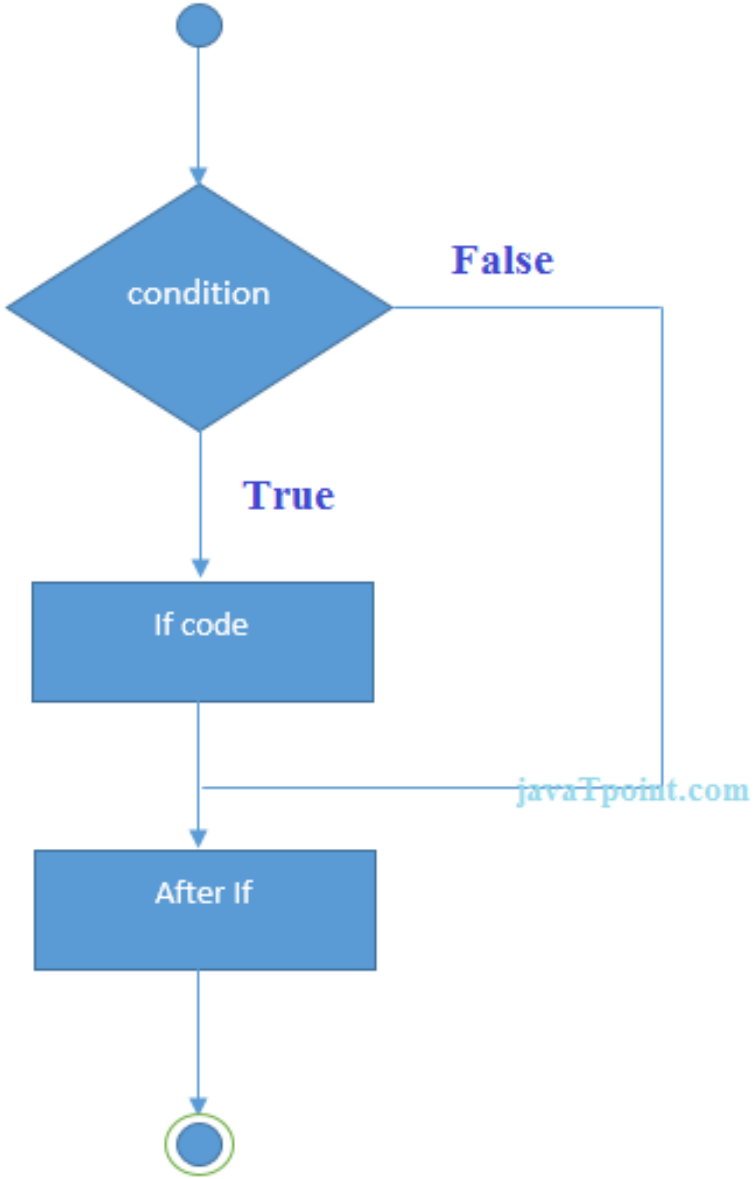
Условен оператор if – else

```
if <условен израз> {
        .....
        .....
        .....
    } else {
        .....
        .....
        .....
}
```

блок 1 от оператори

блок 2 от оператори

1. Условни оператори



1. Условни оператори

Условен оператор if – else if – else

```
if  <условен израз 1> {
        .....
        .....
        .....
    }  else  if <условен израз 2> {
                .....
                .....
                .....
              } else {
                        .....
                        .....
                        .....
            }
```

блок 1 от операции

блок 2 от операции

блок 3 от операции

1. Условни оператори

Вложен оператор if – else

```
if  <условен израз 1> {
        .....
        .....
        .....
    if  <условен израз 2> {
            .....
            .....
            .....
        } else {
                .....
                .....
                .....
        }
}
```

блок 1 от операторите

блок 2 от операторите

блок 3 от операторите

1. Условни оператори

Условен оператор switch – case

```
switch <ключов израз> {
        case value1 :
                ..........
                ..........
                break;
        case value2 :
                ..........
                ..........
                break;
        case value3 :
                ..........
                ..........
                break;
        ..................................................
        default :
                ..........
                ..........
}
```

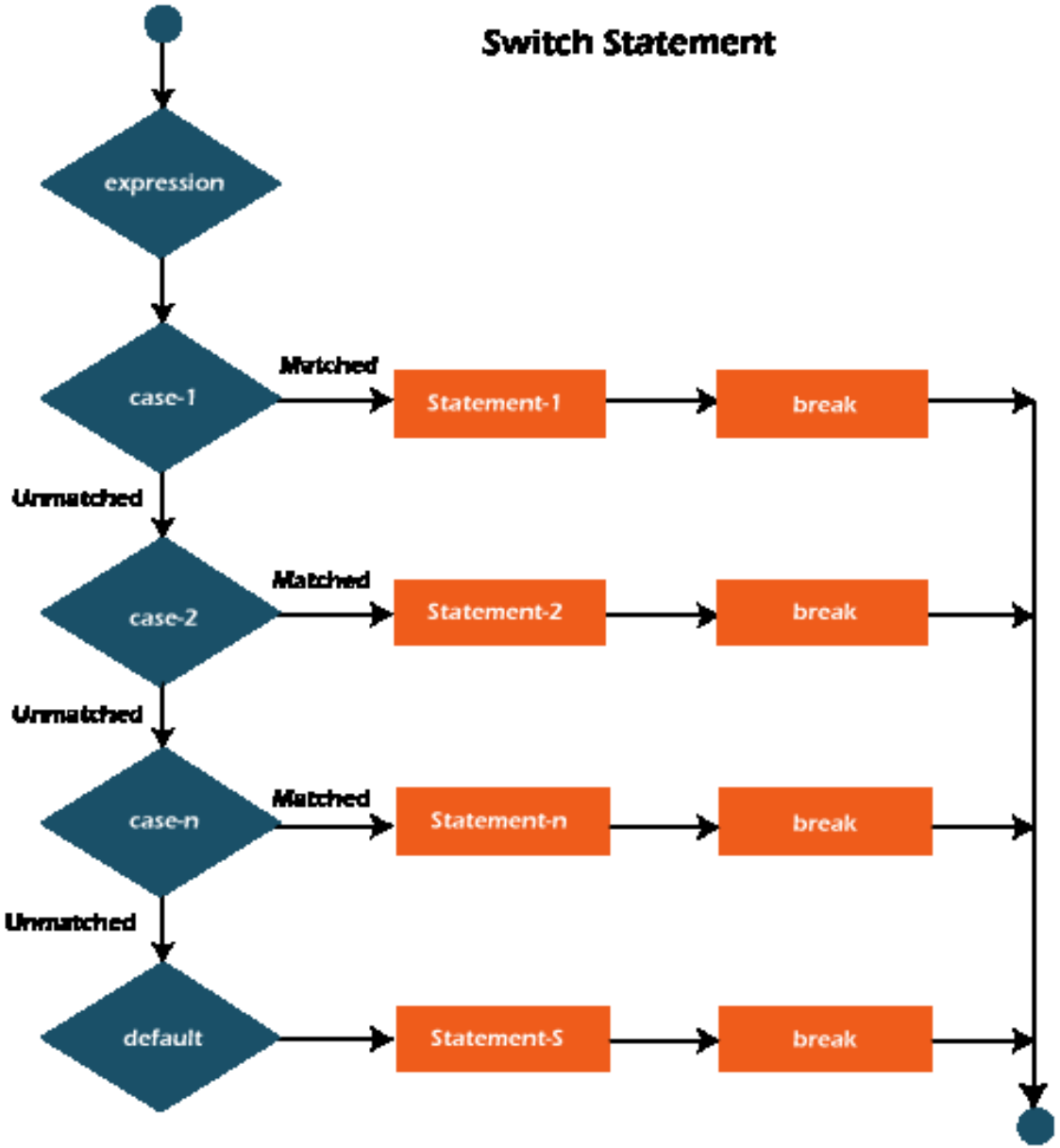
група 1 от оператори

група 2 от оператори

група 3 от оператори

група 4 от оператори

1. Условни оператори



2. Оператори за цикъл

Оператор за цикъл for

for (*begin, condition, increment/decrement*) {

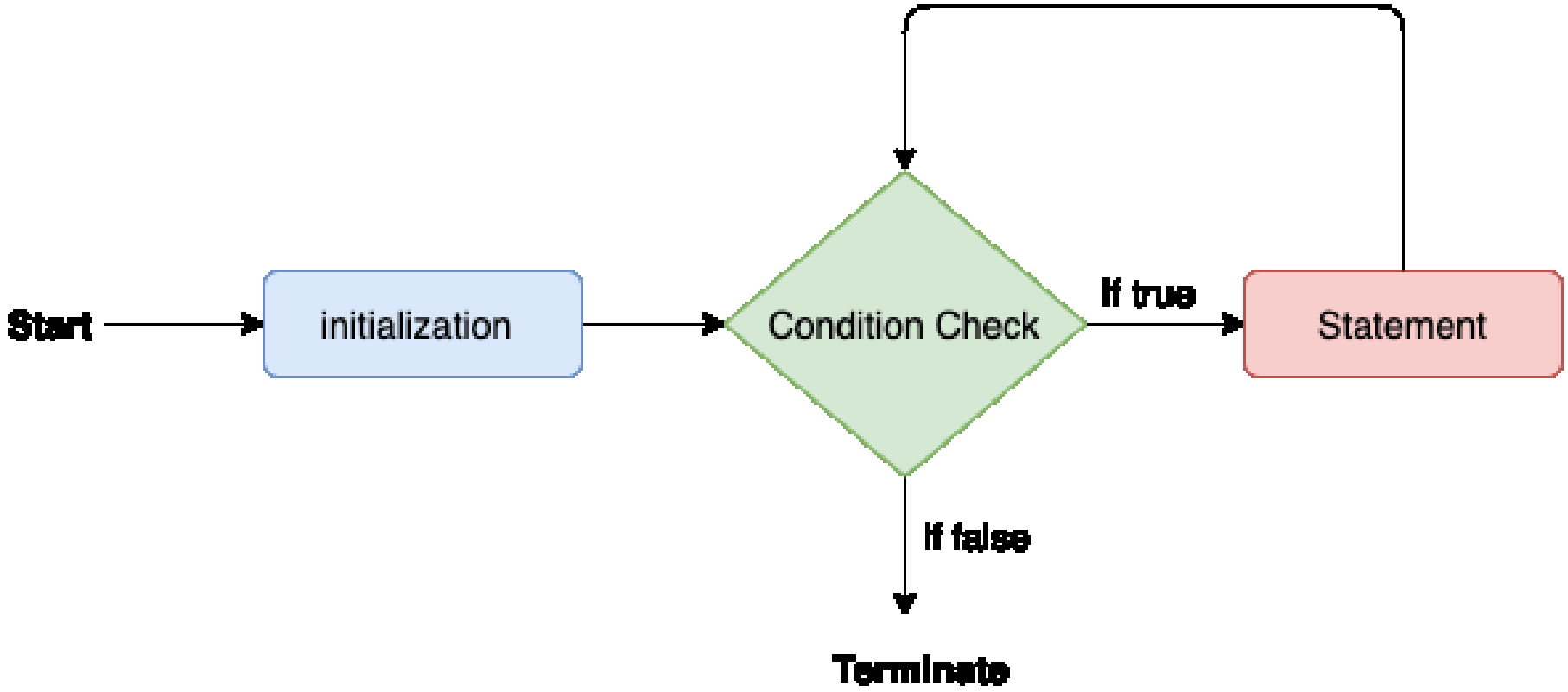
.....

.....

.....

}

блок от оператори



2. Оператори за цикъл

Оператор за цикъл for-each

for (*data_type var:array_name/collection_name*) {

.....

.....

.....

}

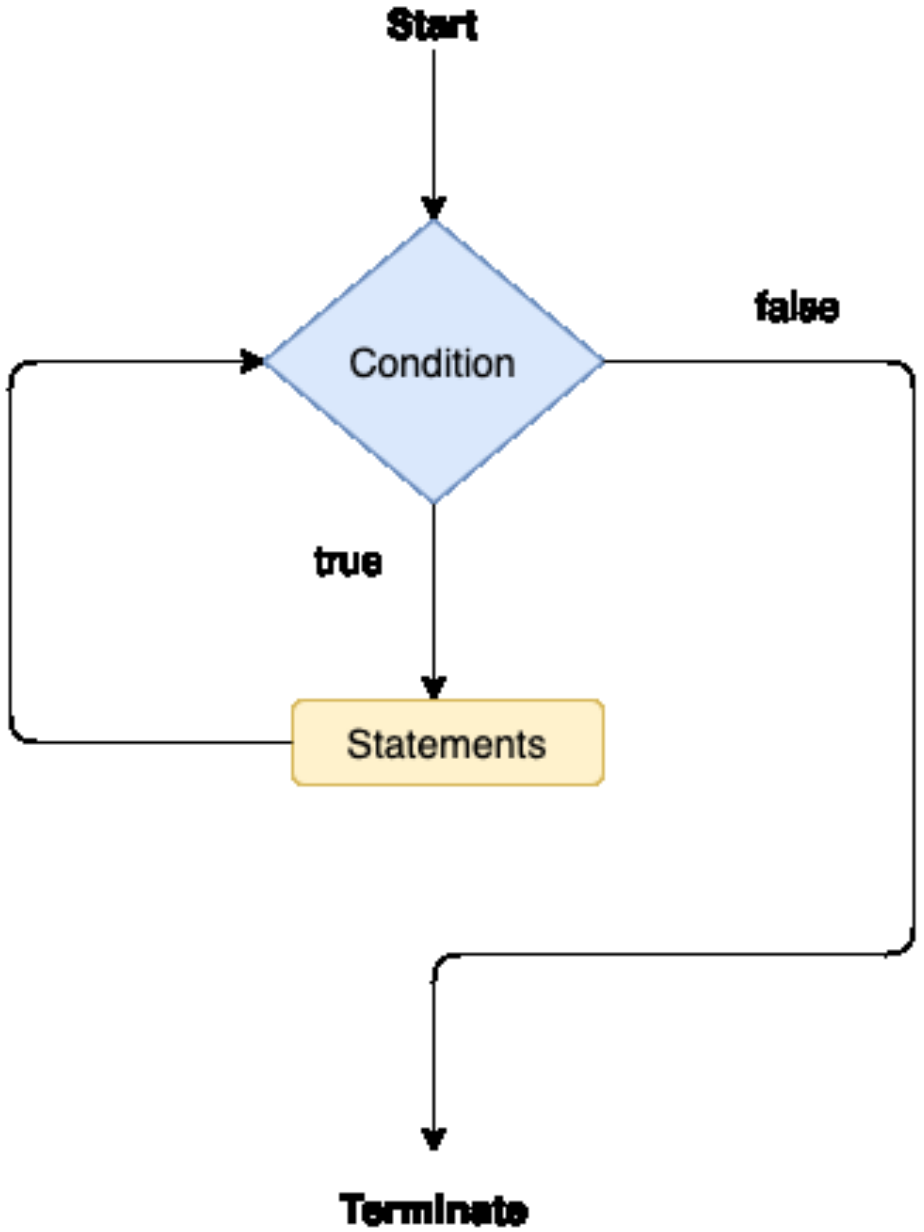
```
public class Calculation {
 public static void main(String[] args) {
  // TODO Auto-generated method stub
   String[] names = {"Java","C","C++","Python","JavaScript"};
   System.out.println("Printing the content of the array:\n");
  for(String name:names) {
        System.out.println(name);
    }
  }
}
```

2. Оператори за цикъл

Оператор за цикъл while

```
while  <условен израз> {
        .....
        .....
        .....
}
```

блок от оператори

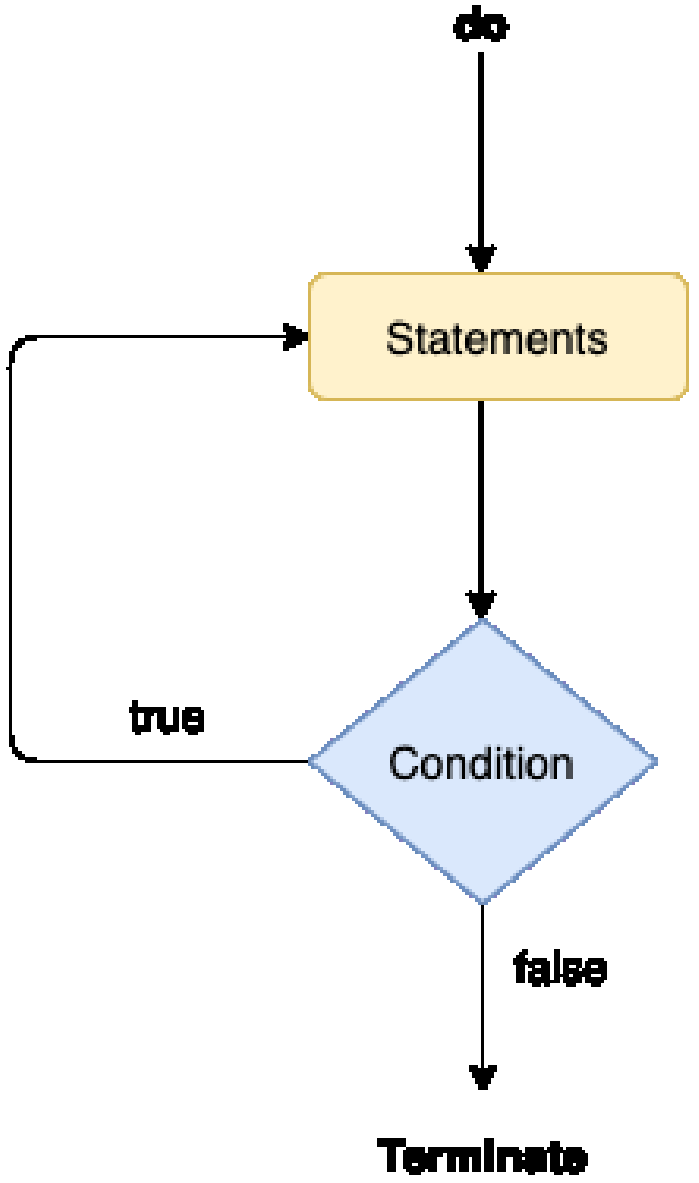


2. Оператори за цикъл

Оператор за цикъл do-while

```
do {
        .....
        .....
        .....
} while <условен израз> ;
```

блок от оператори



2. Оператори за цикъл

Оператори break и continue за прекратяване на изпълнението на цикли.

```
..... {
        .....
        .....
        .....
        break;
        .....
        .....
        .....
}
.....
```

```
..... {
        .....
        .....
        .....
        continue;
        .....
        .....
        .....
}
.....
```

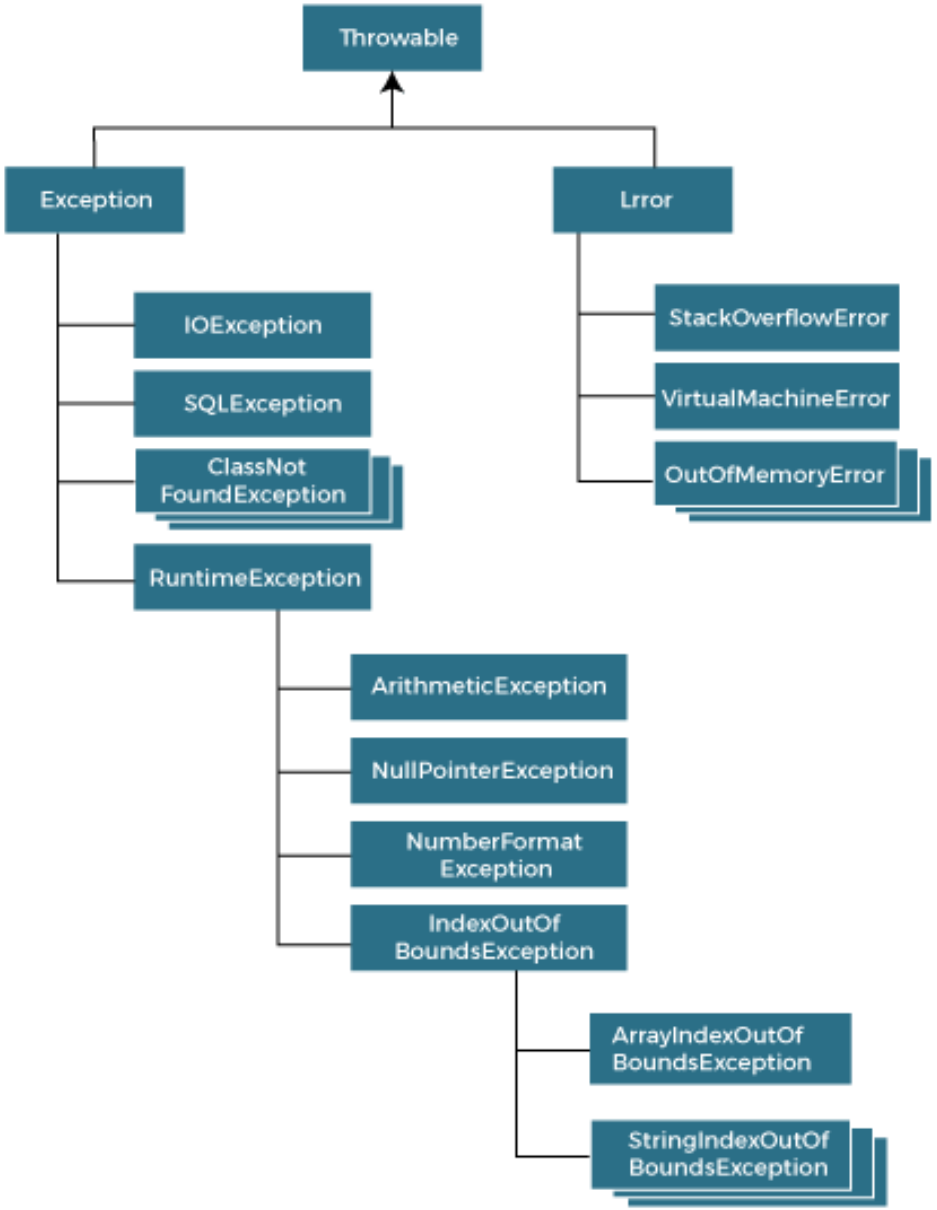
2. Оператори за цикъл

Оператори break и continue за прекратяване на изпълнението на цикли.

```
public class ContinueExample {
  public static void main(String[] args) {
     //for loop
    for ( int i=1; i<=10; i++) {
       if ( i==5 ) {  //using continue statement
           continue;//it will skip the rest statement
        }
       System.out.println(i);
    }
  }
}
```

```
public class BreakExample {
   public static void main(String[] args) {
      //using for loop
     for ( int i=1; i<=10; i++) {
        if ( i==5 ) {
            break;   //breaking the loop
         }
        System.out.println(i);
     }
   }
}
```

3. Изключения



Hierarchy of Java Exception classes

3. Изключения

Keyword	Description
try	Ключовата дума "try" се използва за указване на блок, където трябва да поставим код за изключение. Това означава, че не можем да използваме **try** блока самостоятелно. Блокът **try** трябва да бъде последван от **catch** или **finally**.
catch	Блокът "catch" се използва за обработка на изключението. Той трябва да бъде предшестван от блок **try**, което означава, че не можем да използваме самостоятелно блока **catch**. Той да бъде последван от **finally** блок по-късно.
finally	Блокът "finally" се използва за изпълнение на необходимия код на програмата. Изпълнява се независимо дали е обработено изключение или не.
throw	Ключовата дума "throw" се използва за генериране (хвърляне) на изключение.
throws	Ключовата дума "throws" се използва за деклариране на изключения. Той уточнява, че може да възникне изключение в метода. Не прави изключение. Винаги се използва със сигнатура на метода.

Java Exception Keywords

Видове Java изключения

Има основно два вида изключения: проверени и непроверени. Грешката се счита за непроверено изключение. Въпреки това, според Oracle, има три вида изключения, а именно: проверено изключение, непроверено изключение и грешка.

3. Изключения

Разлика между проверени и непроверени изключения:

- ✓ Проверено изключение (Checked Exception) - Класовете, които директно наследяват класа Throwable, с изключение на RuntimeException и Error, са известни като проверени изключения. Например IOException, SQLException и др. Проверените изключения се проверяват по време на компилиране.
- ✓ Непроверено изключение (Unchecked Exception) - Класовете, които наследяват RuntimeException, са известни като непроверени изключения. Например ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException и др. Непроверените изключения не се проверяват по време на компилиране, но се проверяват по време на изпълнение.
- ✓ Грешка (Error) - Грешката е непоправима. Някои примери за грешки са OutOfMemoryError, VirtualMachineError, AssertionError и др.

3. Изключения

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

            int data=100/0;    //code that may raise e
xception
   } catch ( ArithmeticException e ) {
            System.out.println(e);
   }
            //rest code of the program
   System.out.println("rest of the code...");
  }
}
```

Output:

```
Exception in thread main java.lang.ArithmeticException: / by zero
rest of the code...
```

3. Изключения

Common Scenarios of Java Exceptions (1, 2)

Основните сценарии, при които могат да възникнат непроверени изключения са следните:

- Сценарий, при който възниква ArithmeticException изключение

  Ако разделим някое число на 0 възниква ArithmeticException.

```
int a=50/0;       //ArithmeticException
```

- Сценарий, при който възниква NullPointerException изключение

  Ако имаме нулева стойност в която и да е променлива, извършването на каквато и да е операция върху променливата хвърля NullPointerException.

```
String s=null;
System.out.println(s.length());          //NullPointerException
```

3. Изключения

Common Scenarios of Java Exceptions (3, 4)

- Сценарий, при който възниква NumberFormatException изключение

Ако форматирането на която и да е променлива или число не съответства, това може да доведе до NumberFormatException. Да предположим, че имаме низова променлива, която има знаци; преобразуването на тази променлива в цифра ще предизвика NumberFormatException.

```
String s="abc";
int i=Integer.parseInt(s);        //NumberFormatException
```

- Сценарий, при който възниква ArrayIndexOutOfBoundsException изключение

Когато даден масив надвишава размера си, възниква изключение ArrayIndexOutOfBoundsException. може да има други причини за възникване на ArrayIndexOutOfBoundsException.

```
int a[] = new int[5];
a[10]=50;                    //ArrayIndexOutOfBoundsException
```

3. Изключения

Списък на изключенията в Java:

- Java Try-Catch Block
- Java Multiple Catch Block
- Java Nested Try
- Java Finally Block
- Java Throw Keyword
- Java Exception Propagation
- Java Throws Keyword
- Java Throw vs Throws
- Java Final vs Finally vs Finalize
- Java Exception Handling with Method Overriding
- Java Custom Exceptions