

## 2. Структура программы, типы данных, управляющие конструкции

```
1 package packageB;  
2  
3 public class HelloJava {  
4  
5     public static void main(String[] args) {  
6  
7         System.out.println("Hello Java world!");  
8  
9     }  
10 }
```

## Объявление класса

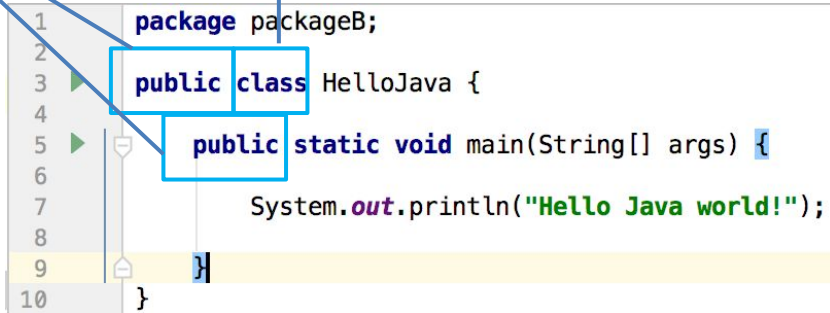


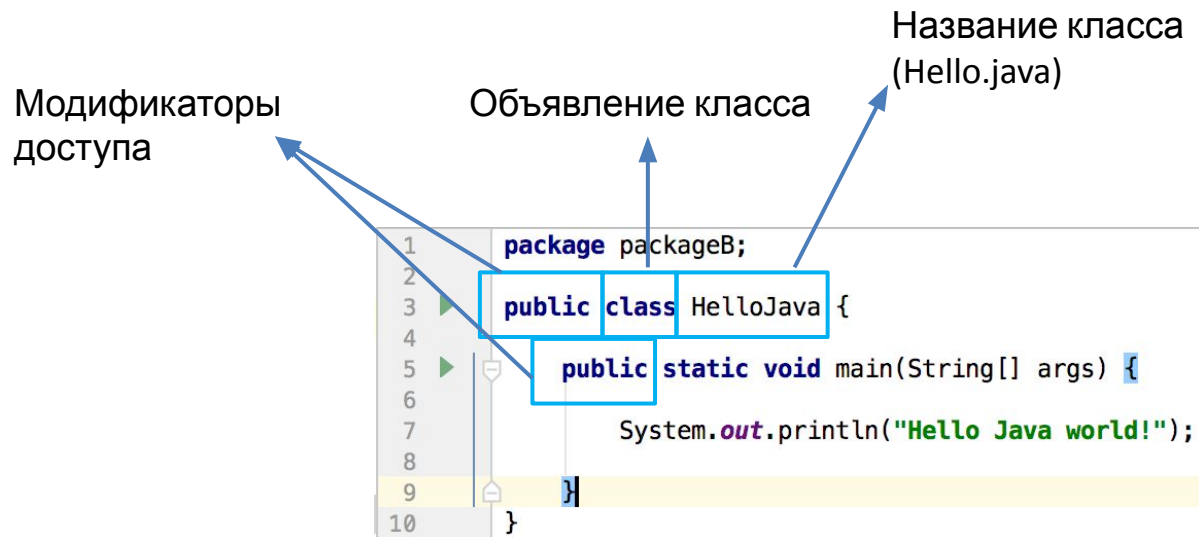
```
1 package packageB;  
2  
3 public class HelloJava {  
4  
5     public static void main(String[] args) {  
6  
7         System.out.println("Hello Java world!");  
8  
9     }  
10 }
```

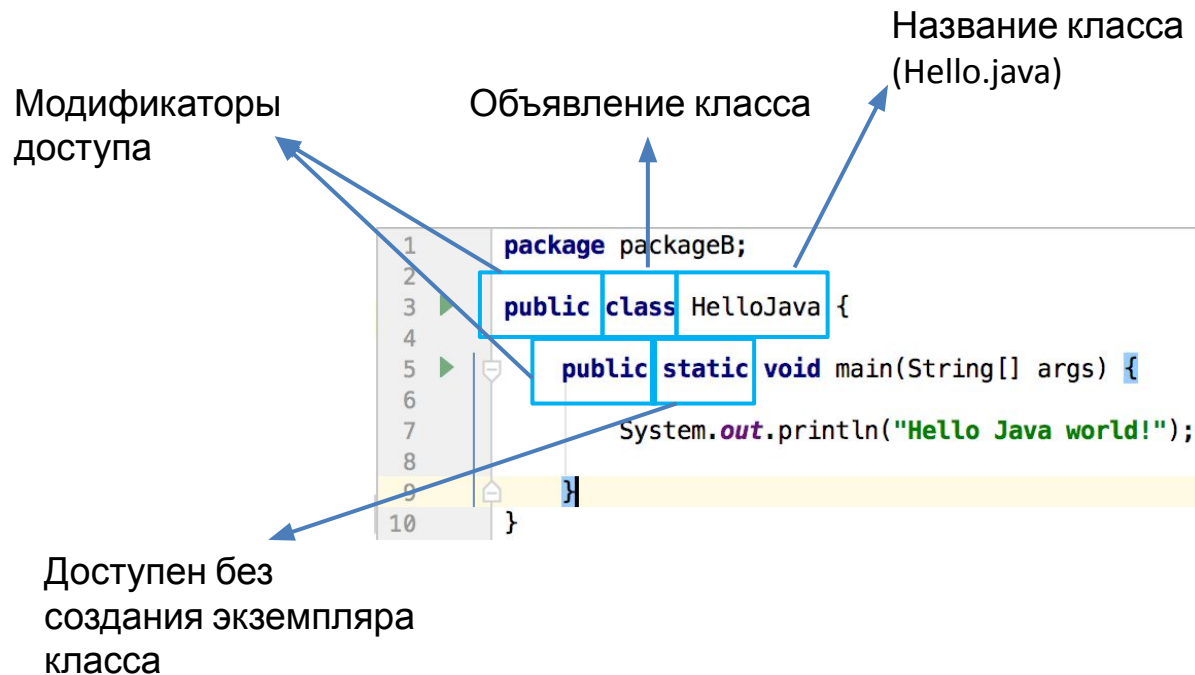
Модификаторы  
доступа

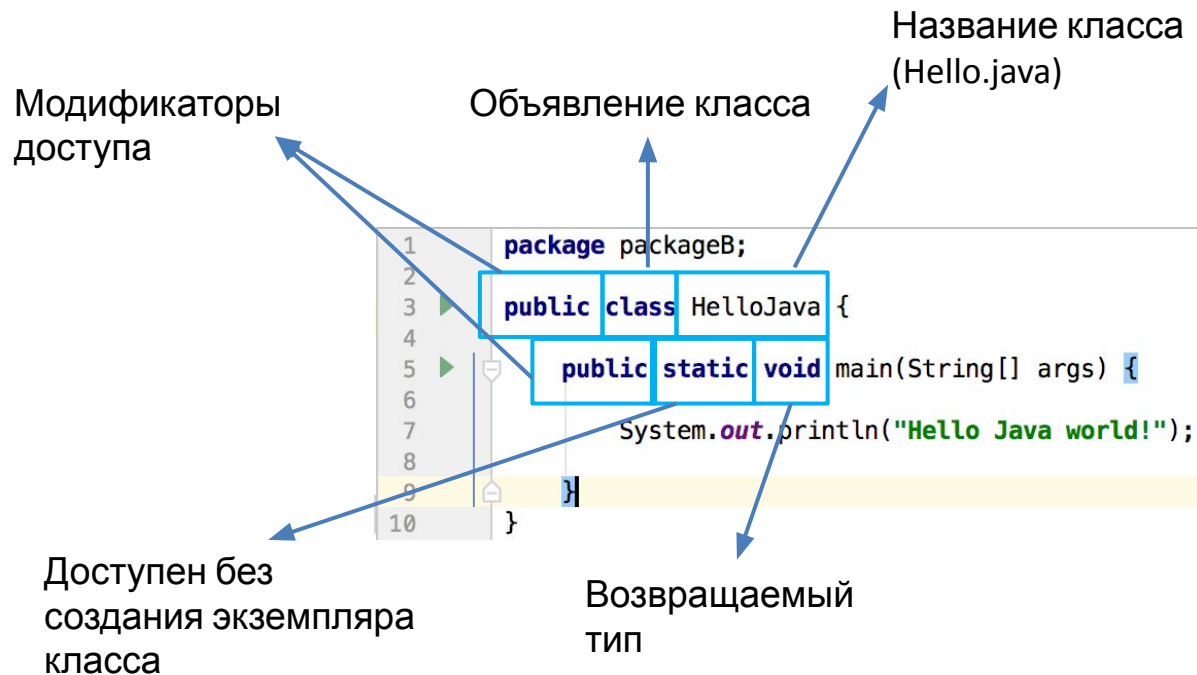
Объявление класса

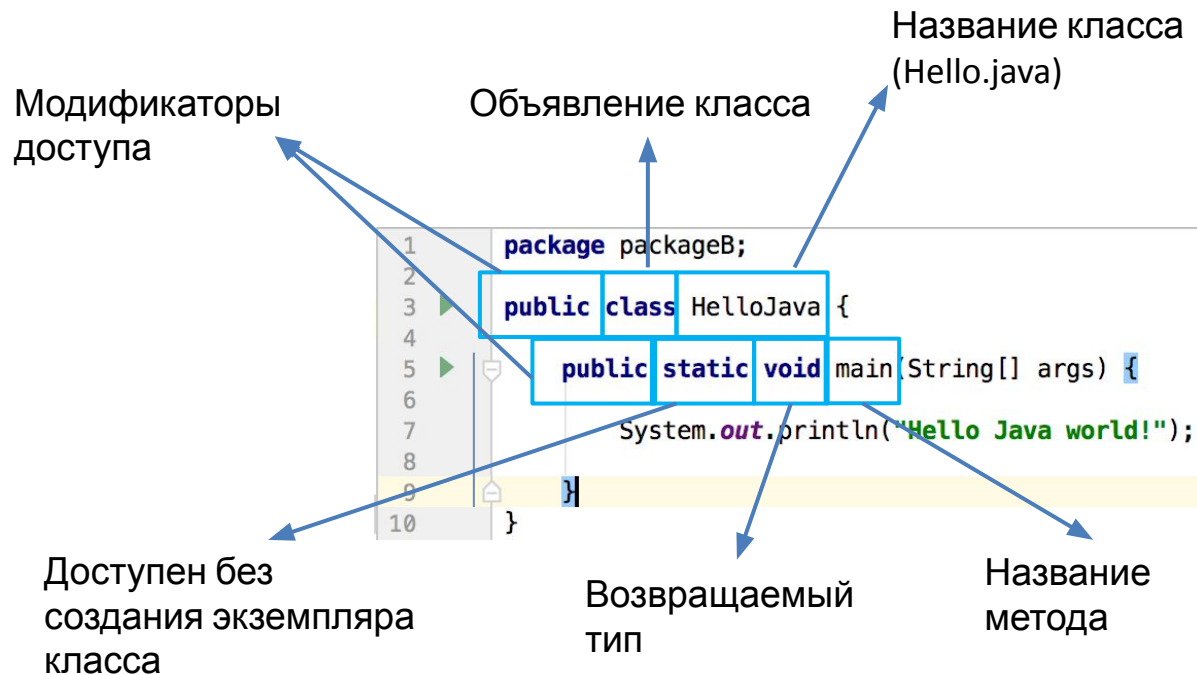
```
1 package packageB;  
2  
3 public class HelloJava {  
4  
5     public static void main(String[] args) {  
6  
7         System.out.println("Hello Java world!");  
8  
9     }  
10 }
```



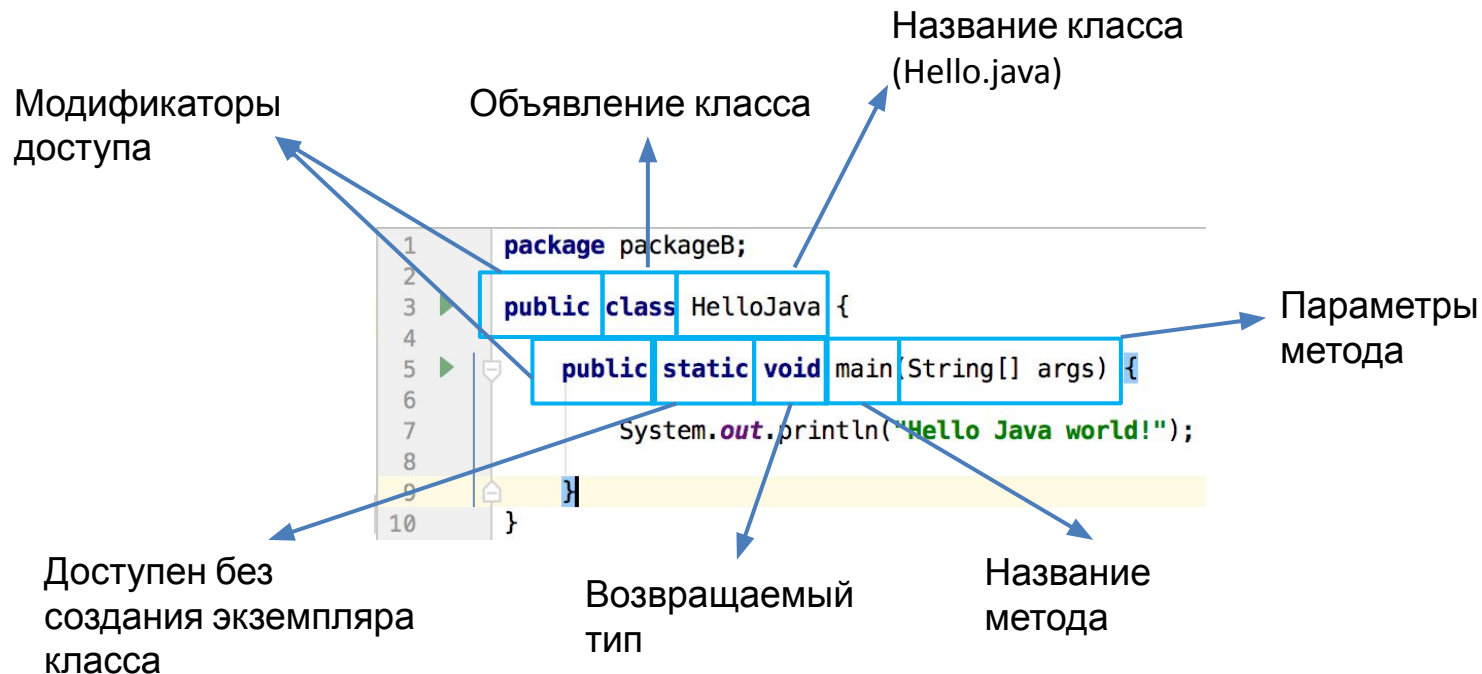


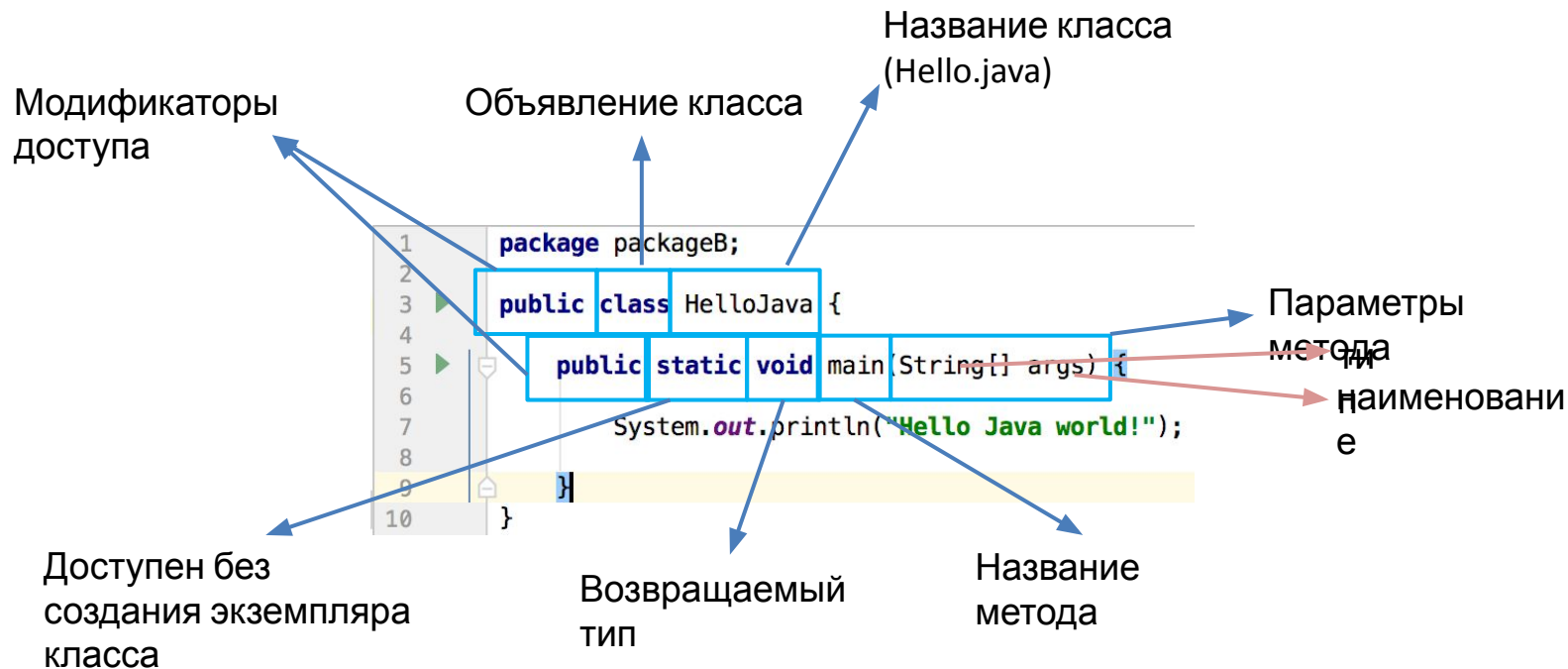


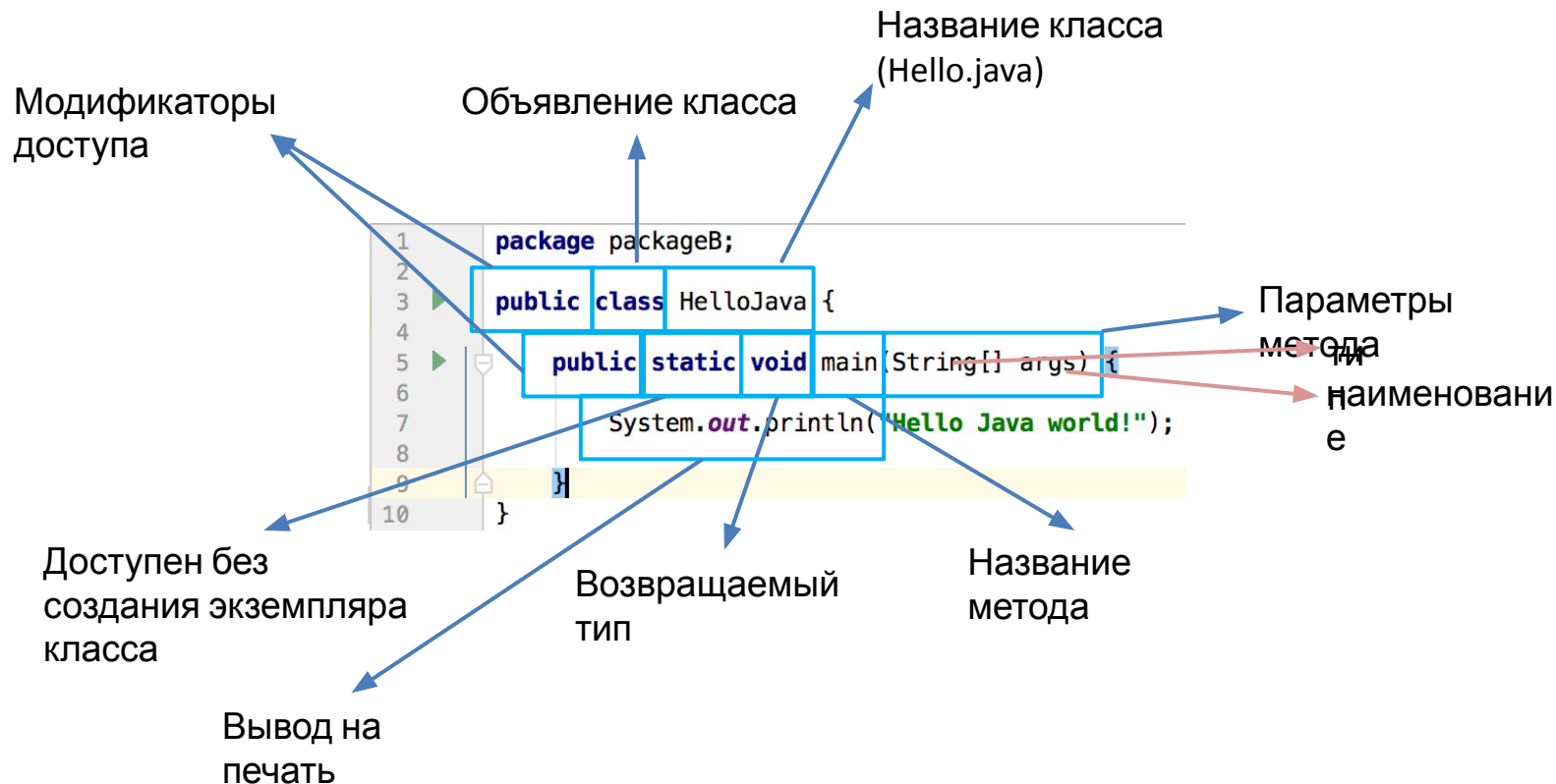


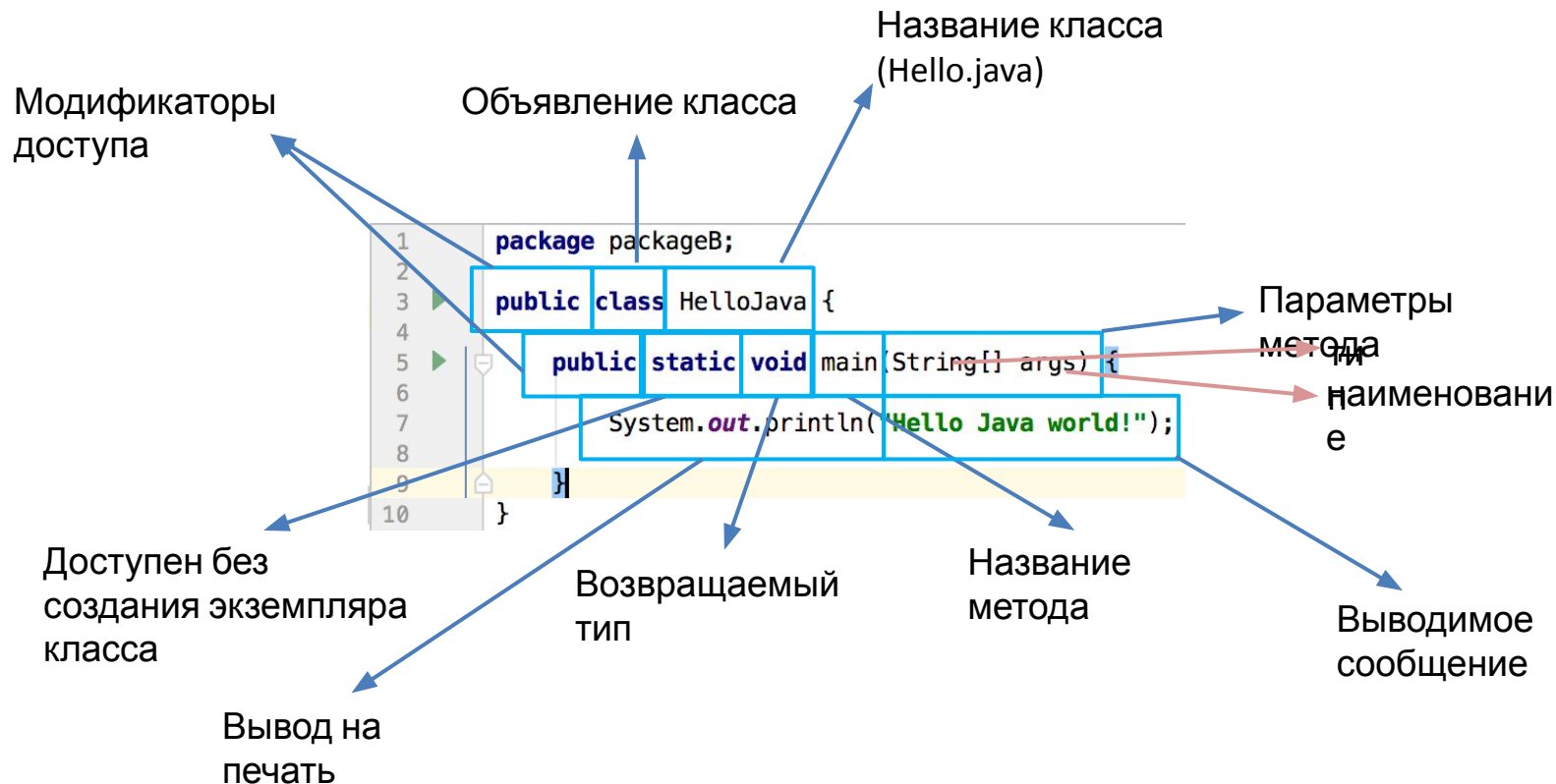


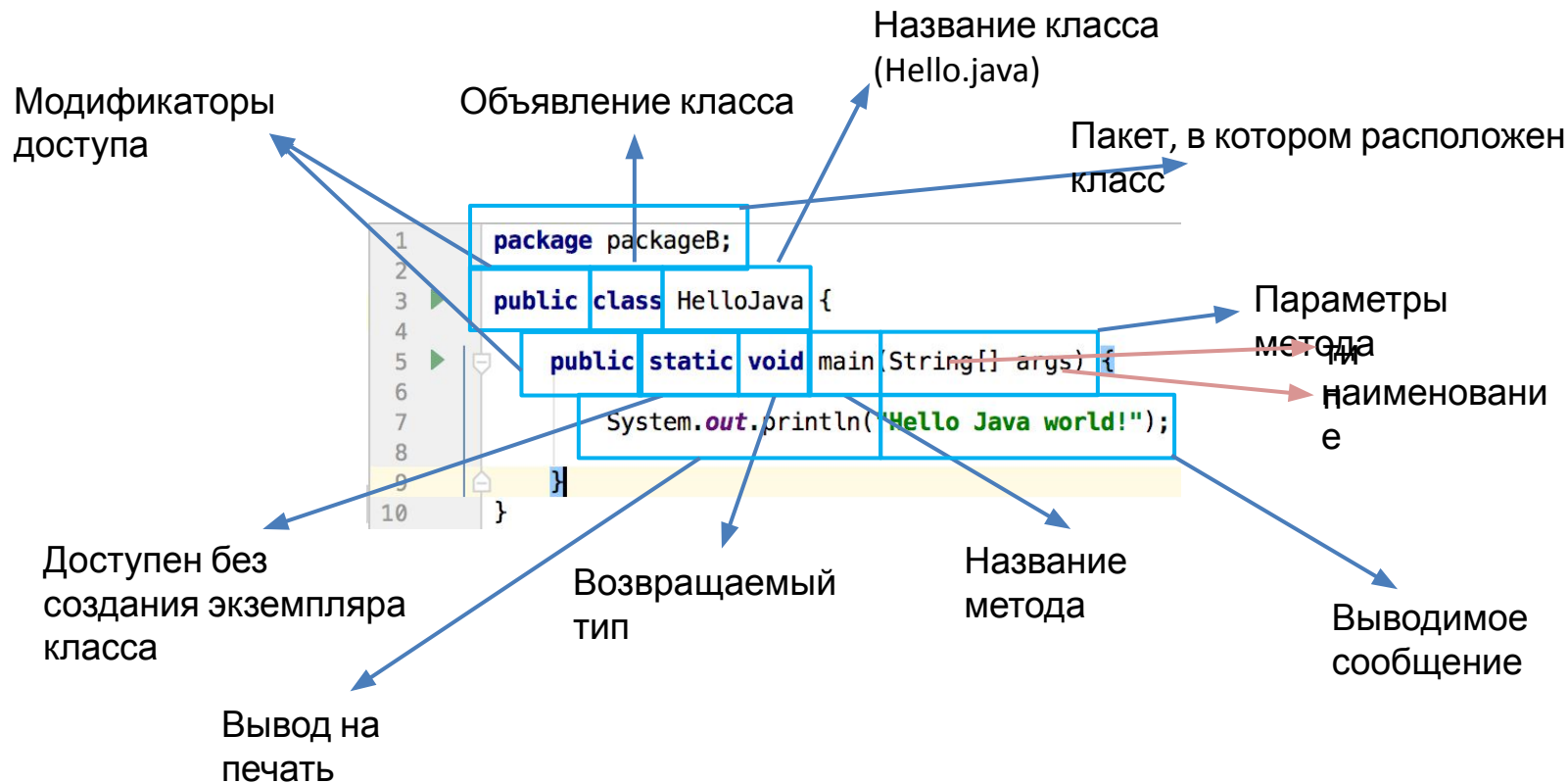


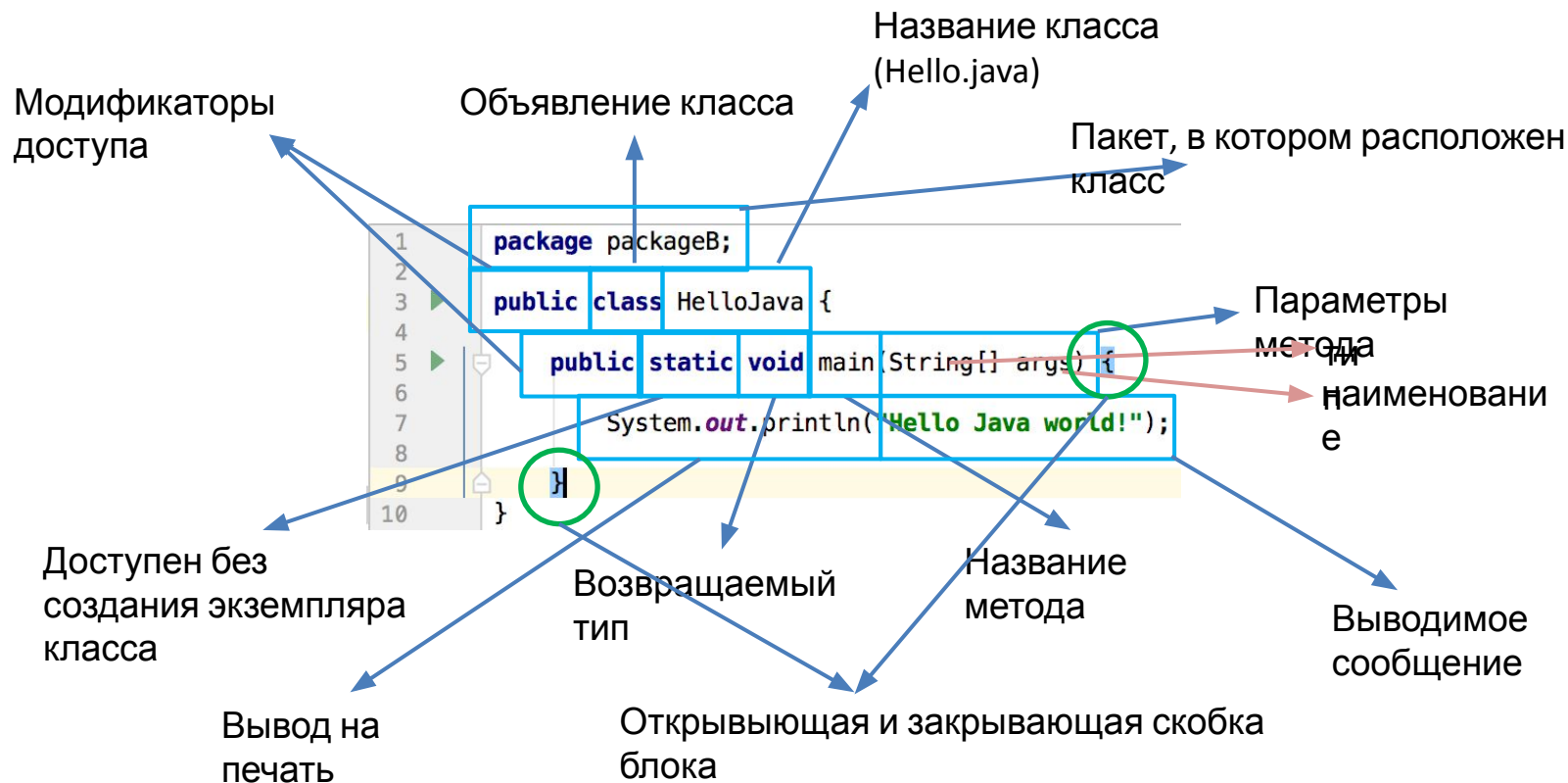




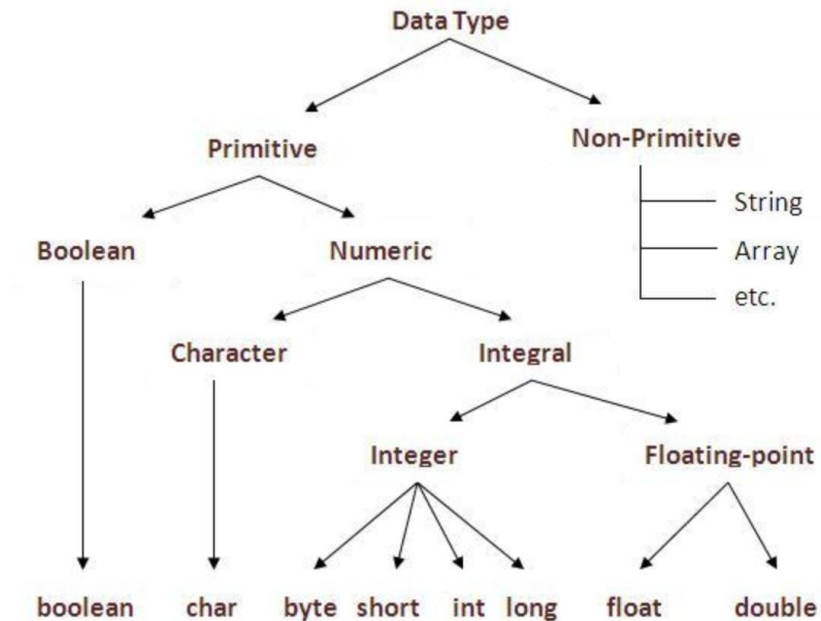








- Прimitives
- Non-Primitives



| Тип     | Значение по умолчанию | Размер |
|---------|-----------------------|--------|
| boolean | false                 | 1 byte |
| char    | '\u0000'              | 2 byte |
| byte    | 0                     | 1 byte |
| short   | 0                     | 2 byte |
| int     | 0                     | 4 byte |
| long    | 0                     | 8 byte |
| float   | 0.0f                  | 4 byte |
| double  | 0.0d                  | 8 byte |



```
public class PrimitiveDemo1 {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 30;  
        int sum = a + b;  
        System.out.println(sum); // 40  
    }  
}
```

```
public class PrimitiveDemo2 {  
    public static void main(String[] args) {  
        short year1 = 25;  
        short year2 = 50;  
        System.out.println(year1 + year2); // 75  
    }  
}
```

```
public class PrimitiveDemo3 {  
    public static void main(String[] args) {  
        long l = 100_000_000;  
        int i = 50000;  
        //int sum = l + i; // Compile error  
        long sum = l + i;  
        System.out.println(sum); // 100050000  
    }  
}
```

```
public class PrimitiveDemo4 {  
    public static void main(String[] args) {  
        float f = 234.9f;  
        double d = 678.8;  
        //float sum = f + d; // Compile error  
        //byte sum = (byte) (f + d);  
        double sum_double = f + d;  
        int sum_int = (int) (f + d);  
        System.out.println(sum_double); // 913.6999938964843  
        System.out.println(sum_int); // 913  
    }  
}
```

```
public class PrimitiveDemo5 {  
    public static void main(String[] args) {  
        byte b = 127;  
        //    b = 128; // 127 - max  
        short s = 32767;  
        //    s = 32768; // 32767 - max  
        long l = 123_456_789;  
        l = 123456789l;  
        l = 123_456_789L;  
        l = 0xAAD1; // 43729  
        l = 027; // 23  
        l = 0b1001_0101_1001_1011; // 38299  
        l = 'A'; // 65;  
        //    float f = 23.45; // Compile error  
        float f = 23.45f;  
        double d = 11.33;  
        char c = (char) 65; // A  
        boolean bool = true;  
        //    bool = "false"; // Compile error  
        //    bool = 0; // Compile error  
    }  
}
```

| Тип оператора         | Категория                 | Выражение                                                              |
|-----------------------|---------------------------|------------------------------------------------------------------------|
| Унарный               | Постфиксная               | <i>expr++ expr--</i>                                                   |
|                       | Префиксная                | <i>++expr --expr +expr -expr ~ !</i>                                   |
| Арифметический        | Мультипликативная         | <i>* / %</i>                                                           |
|                       | Сложение/вычитание        | <i>+ -</i>                                                             |
| Сдвиг                 | Сдвиг                     | <i>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</i>                                  |
| Операция отношения    | Сравнения                 | <i>&lt; &gt; &lt;= &gt;= instanceof</i>                                |
|                       | Равенства                 | <i>== !=</i>                                                           |
| Побитовый             | Побитового И              | <i>&amp;</i>                                                           |
|                       | Исключающее побитовое ИЛИ | <i>^</i>                                                               |
|                       | Включающее побитовое ИЛИ  | <i> </i>                                                               |
| Логический            | Логическое И              | <i>&amp;&amp;</i>                                                      |
|                       | Логическое ИЛИ            | <i>  </i>                                                              |
| Тернарный             | Тернарный                 | <i>? :</i>                                                             |
| Оператор присваивания | Присваивание              | <i>= += -= *= /= %= &amp;= ^=  = &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;=</i> |

## Пример арифметических операций

```
public class OperatorsDemo1 {  
    public static void main(String[] args) {  
        System.out.println(10*10/5+3-1*4/2); // 21  
    }  
}
```

## Пример операций с присваиванием

```
public class OperatorsDemo2 {  
    public static void main(String[] args) {  
        int x = 10;  
        System.out.println(x++); // 10  
        System.out.println(++x); // 12  
        System.out.println(x--); // 12  
        System.out.println(--x); // 10  
    }  
}
```

## Управляющие конструкции – позволяют направить работу программы по одному из путей в зависимости от определенных условий

- if..else
- switch..case

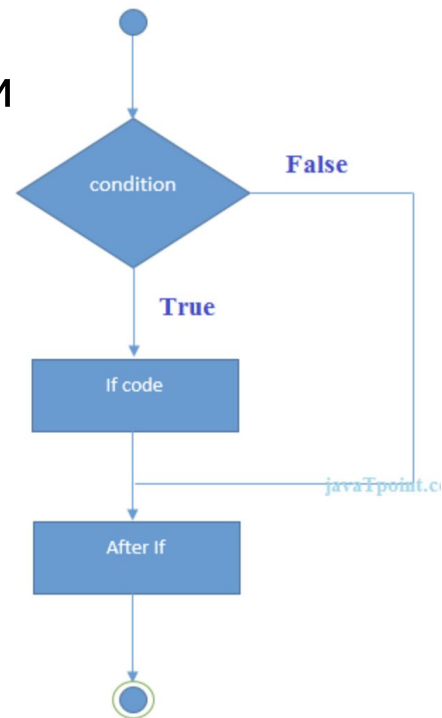
```
if(condition){  
    //code to be executed  
}
```

```
switch(expression){  
    case value1:  
        //code to be executed;  
        break; //optional  
    case value2:  
        //code to be executed;  
        break; //optional  
    .....  
    default:  
        code to be executed if all cases are not matched;  
}
```

**If/else** – условный оператор, который проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код

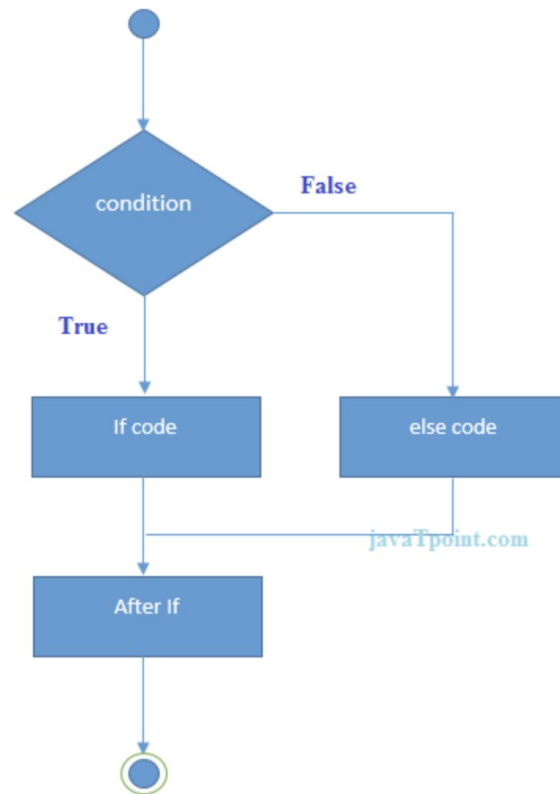
```
if (condition) {  
    // do something  
}
```

```
public class IfElseDemo1 {  
    public static void main(String[] args) {  
        String cardType = "VISA";  
        if ("VISA".equals(cardType)) {  
            System.out.println("You pay with VISA ");  
        }  
        System.out.println("done");  
    }  
}
```



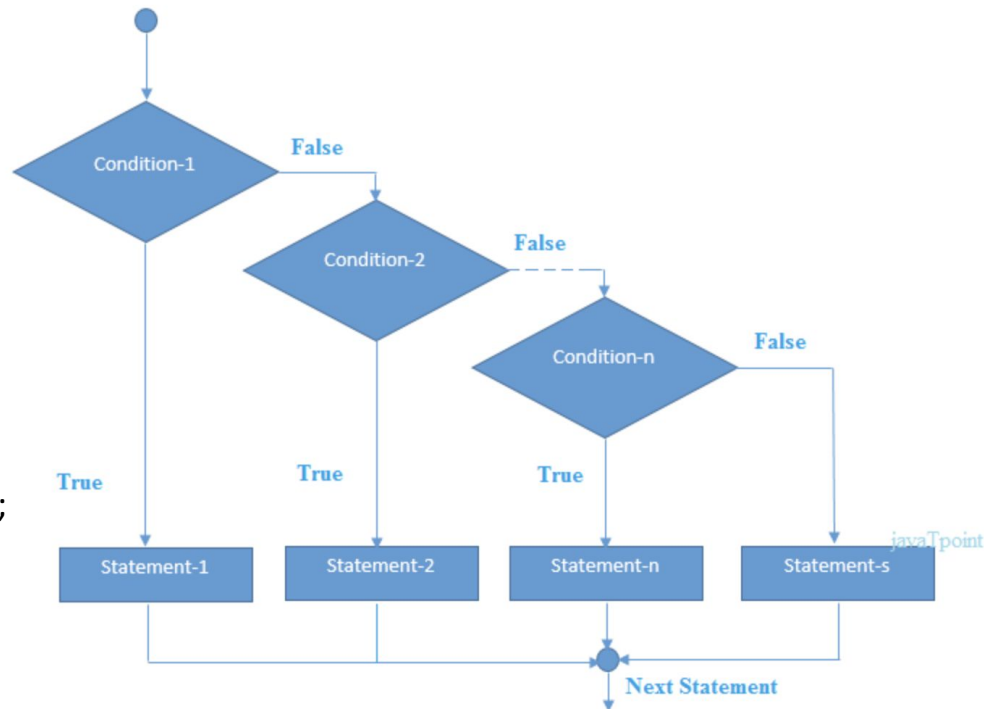
```
if (condition) {  
    // do something if condition is TRUE  
} else {  
    // do something if condition is FALSE  
}
```

```
public class IfElseDemo2 {  
    public static void main(String[] args) {  
        String cardType = "VISA";  
        if ("VISA".equals(cardType)) {  
            System.out.println("You pay with VISA");  
        } else {  
            System.out.println("Unknown card");  
        }  
        System.out.println("done");  
    }  
}
```



```
if (condition1) {
    // do something if condition1 is TRUE
} else if (condition2) {
    // do something if condition2 is TRUE
} else if (condition3) {
    // do something if condition3 is TRUE
} else {
    // do something here if all the conditions are false
}
```

```
public class IfElseDemo3 {
    public static void main(String[] args) {
        String cardType = "VISA";
        if ("VISA".equals(cardType)) {
            System.out.println("You pay with VISA");
        } else if ("Mastercard".equals(cardType)) {
            System.out.println("You pay with Mastercard");
        } else {
            System.out.println("Unknown card");
        }
        System.out.println("done");
    }
}
```

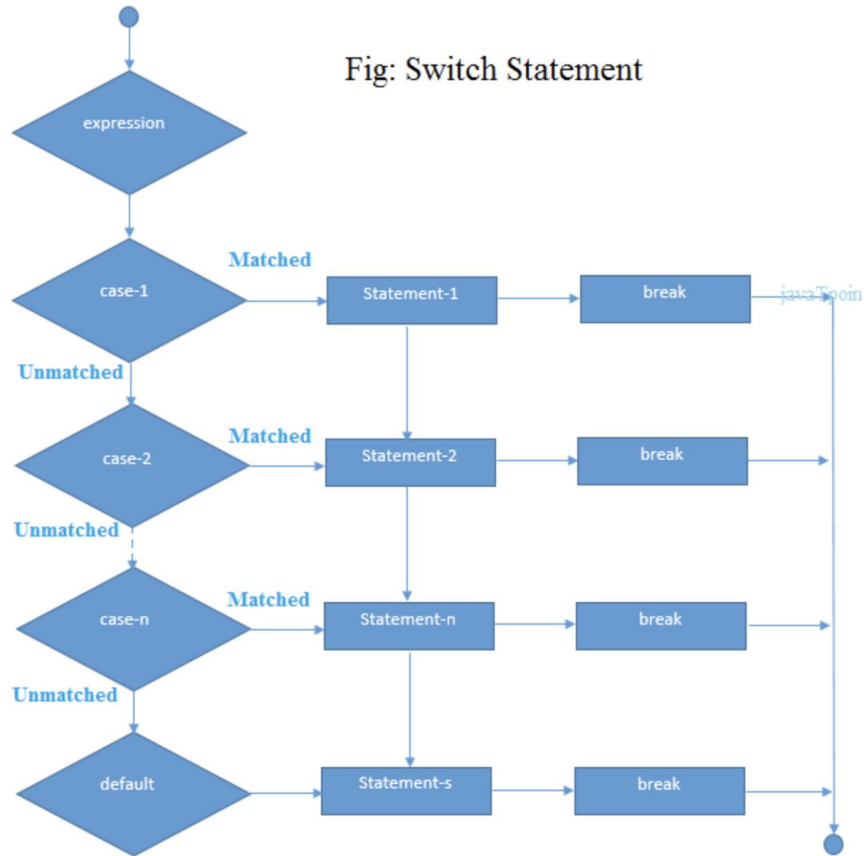




**switch/case** – условный оператор, аналогичный if/else + позволяет обработать несколько условий

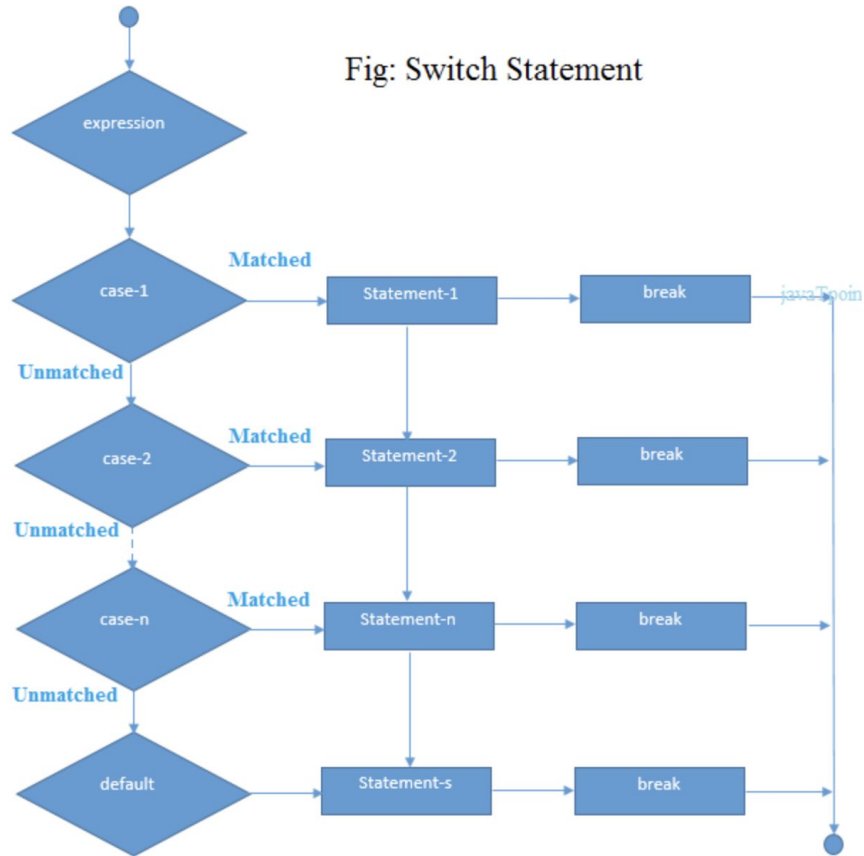
```
switch (expression) {  
    case (value1):  
        // code if expression == value1  
        break;  
    case (value2):  
        // code if expression == value2  
        break;  
    default:  
        // code if default  
}
```

Fig: Switch Statement



```
public class SwitchCaseDemo1 {
    public static void main(String[] args) {
        String cardType = "VISA";
        switch (cardType) {
            case "VISA":
                System.out.println("You pay with VISA");
                break;
            case "Mastercard":
                System.out.println("You pay with Mastercard");
                break;
            default:
                System.out.println("Unknown card");
        }
        System.out.println("done");
    }
}
```

Fig: Switch Statement



# Тернарный оператор – сокращенная запись if/else

```
if (condition) {  
    // do something if condition is TRUE  
} else {  
    // do something if condition is FALSE  
}
```

```
public class IfElseOperatprDemo1 {  
    public static void main(String[] args) {  
        String cardType = "VISA";  
        String result = cardType.equals("VISA")  
            ? "You pay with VISA"  
            : "Unknown card";  
        System.out.println(result);  
    }  
}
```

**Массив** – объект, хранящий в себе фиксированное количество значений одного типа



```
public class ArrayDemo1 {  
    public static void main(String[] args) {  
        int[] intArray = new int[15];  
        intArray[0] = 10;  
        intArray[1] = 20;  
        intArray[2] = 30;  
  
        System.out.println(Arrays.toString(intArray));  
    }  
}
```

```
public class ArrayDemo2 {  
    public static void main(String[] args) {  
        int[] intArray = {1,2,3,4,5,6};  
  
        System.out.println(Arrays.toString(intArray));  
    }  
}
```

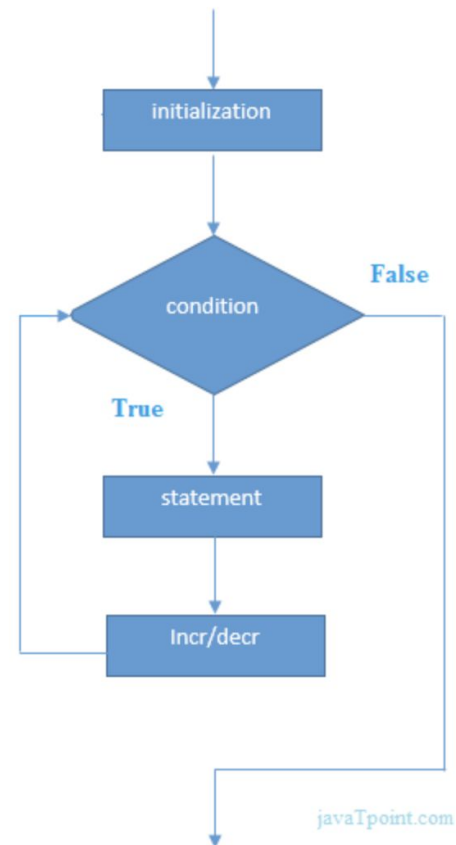
**Цикл** – позволяет в зависимости от определенных условий выполнять определенное действие множество раз

- for
- while
- do...while

## for – заранее известно количество итераций

```
1 for ([инициализация счетчика]; [условие]; [изменение счетчика])
2 {
3     // действия
4 }
```

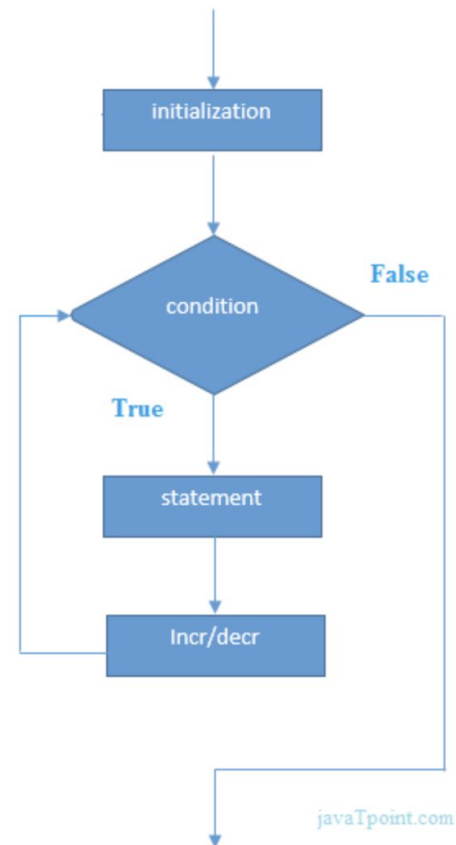
```
public class ForDemo1 {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            System.out.println(i);
        }
    }
}
```



## for – заранее известно количество итераций

```
1 for ([инициализация счетчика]; [условие]; [изменение счетчика])
2 {
3     // действия
4 }
```

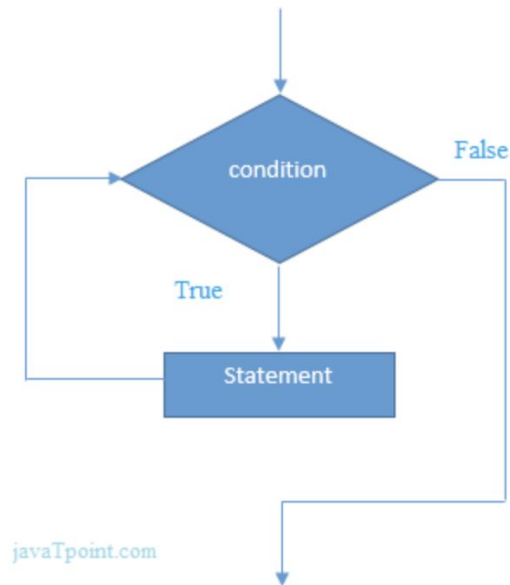
```
public class ForDemo2 {
    public static void main(String[] args) {
        int[] intArray = new int[10];
        for (int i = 0; i < 10; i++) {
            intArray[i] = i * 2;
        }
        System.out.println(Arrays.toString(intArray));
    }
}
```





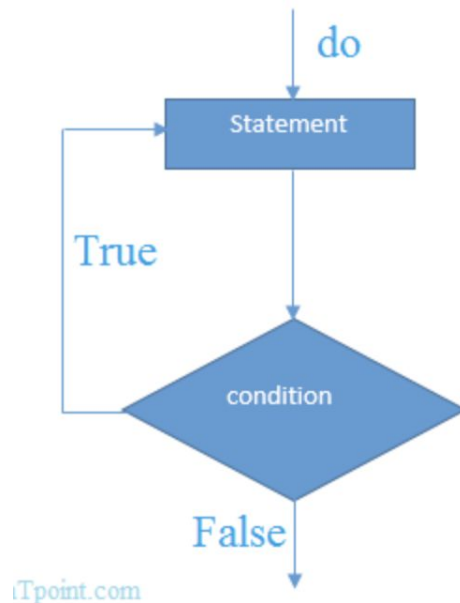
**while** - сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется

```
public class WhileDemo1 {  
    public static void main(String[] args) {  
        int i = 0;  
        while (i < 10) {  
            System.out.println(i++);  
        }  
    }  
}
```



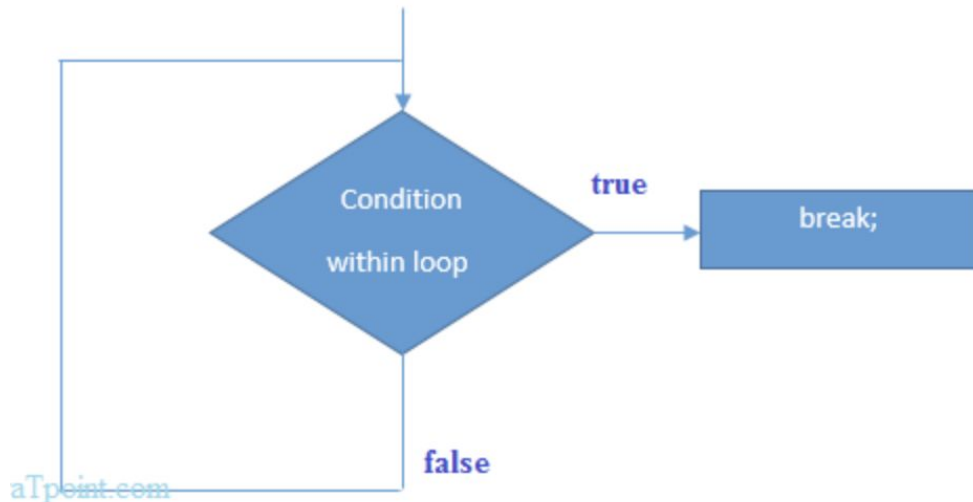
**do...while** – сначала выполняется код цикла, а затем проверяется условие

```
public class DoWhileDemo1 {  
    public static void main(String[] args) {  
        int i = 0;  
        do {  
            System.out.println(i++);  
        } while (i < 10);  
    }  
}
```



**break** – прерывает выполнение цикла и выходит из него

```
public class BreakDemo1 {  
    public static void main(String[] args) {  
        for (int count = 10; count > 0; count--) {  
            System.out.println("count: " + count);  
  
            for (int i = 1; i <= 10; i++) {  
                System.out.println("i: " + i);  
                if (count == i) {  
                    break;  
                }  
            }  
        }  
    }  
}
```



**continue** – переходит сразу же к следующей итерации цикла

```
public class ContinueDemo1 {  
    public static void main(String[] args) {  
        for (int count = 0; count < 10; count++) {  
            System.out.println("count: " + count);  
  
            for (int i = 1; i <= 10; i++) {  
                System.out.println("i: " + i);  
                if (count == i) {  
                    break;  
                }  
            }  
        }  
    }  
}
```