

Java and OOP Basics

Quality Engineering Studio
Bogotá - Colombia

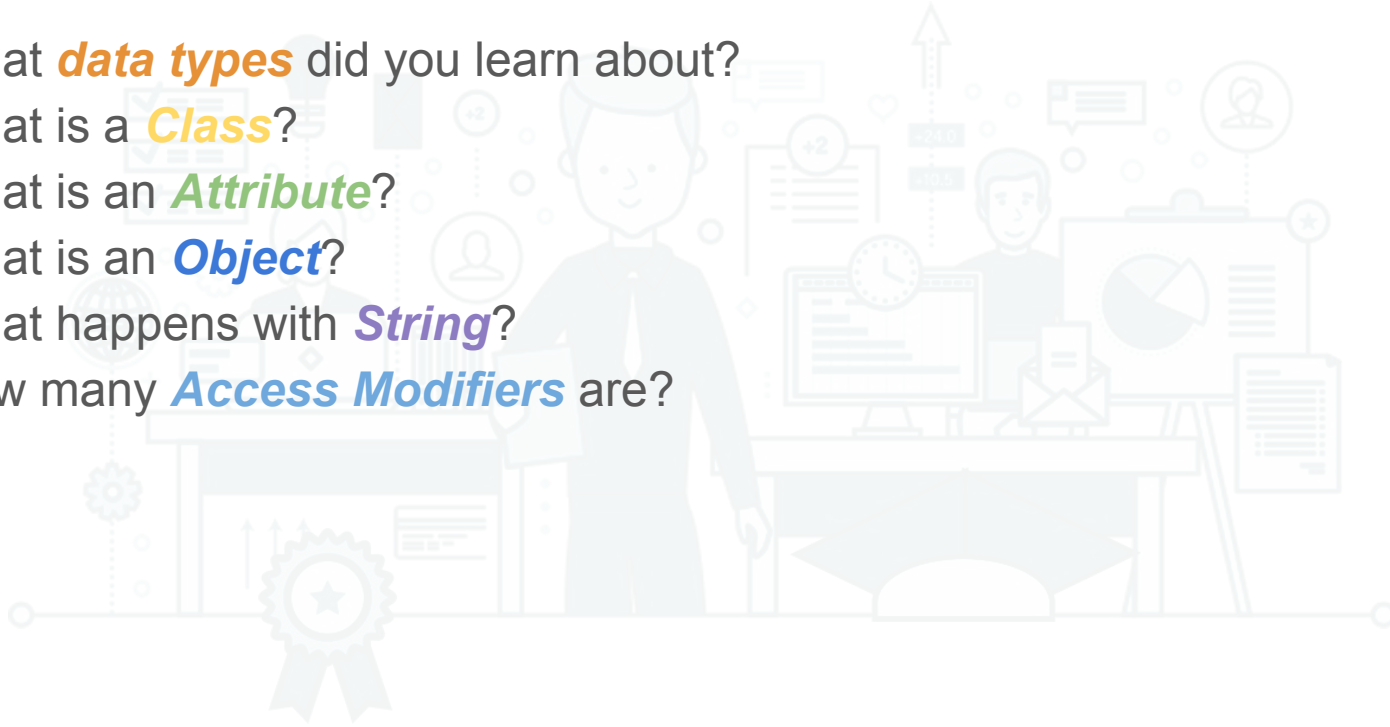
Agenda

- Last session
- Package
- Class structure
- Methods
- Conditionals
- Loops
- Exercises



Last Session

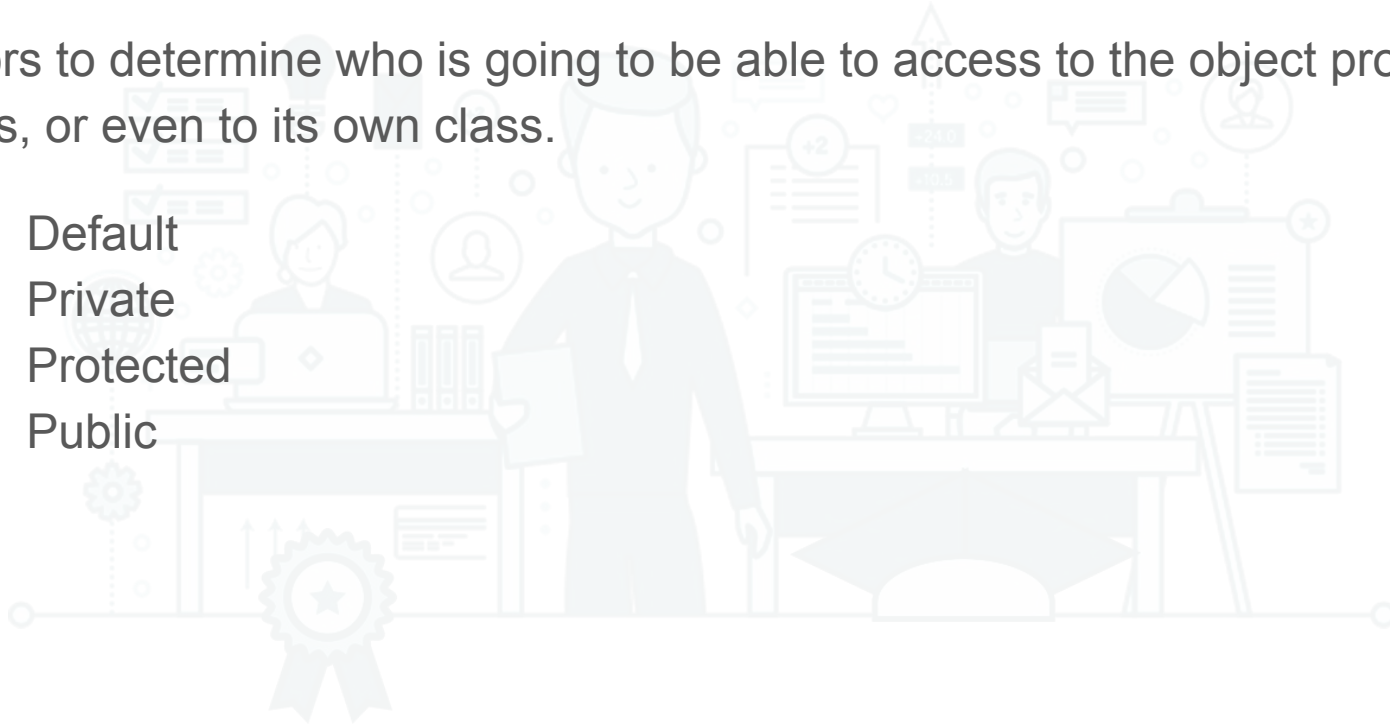
- What **data types** did you learn about?
- What is a **Class**?
- What is an **Attribute**?
- What is an **Object**?
- What happens with **String**?
- How many **Access Modifiers** are?



Access Modifiers

Indicators to determine who is going to be able to access to the object properties, methods, or even to its own class.

- Default
- Private
- Protected
- Public



Access Modifier

Access Modifier	Class	Package	Subclass	All
Default	Yes	Yes	No	No
Private	Yes	No	No	No
Protected	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes

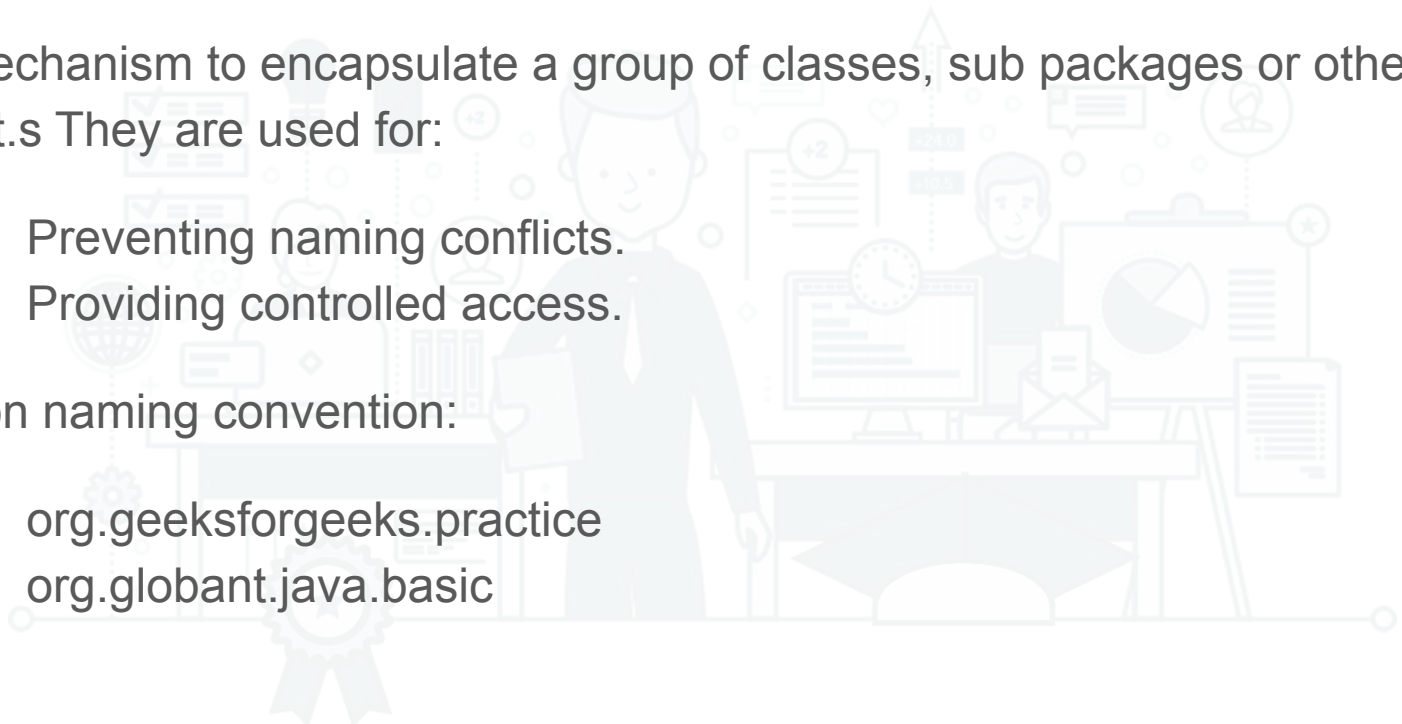
Package

It's a mechanism to encapsulate a group of classes, sub packages or other elements. They are used for:

- Preventing naming conflicts.
- Providing controlled access.

Common naming convention:

- `org.geeksforgeeks.practice`
- `org.globant.java.basic`



Class structure

```
package com.globant.java.basic;
```

```
import java.util.Date;
```

```
public class Person{
```

```
    private String name = "Andres";
```

```
    private int age = 25;
```

```
    private Date birthDate = new Date("11/14/1993");
```

```
    public void updateInformation(String newName, int newAge, Date newBirthDate){
```

```
        this.name = newName;
```

```
        this.age = newAge;
```

```
        this.birthDate = newBirthDate;
```

```
    }
```

```
    public String printInformation(){
```

```
        String name = ("Name: " + this.name + "\nAge: " + this.age + "\nBirth date: " +  
        this.birthDate);
```

```
        return name ;
```

```
    }
```

```
}
```

Methods

Algorithms to define what an object can do, implying issues like object instantiation, setting properties, getting properties, etc.

Form:

```
<Access Modifier> <Return data type> <name> (<parameters>) {  
    <Method body>  
}
```


Examples

```
public int sumNumbers(int num1, int num2){  
    return num1 + num2;  
}
```

```
private int num;
```

```
public void initializeVariable(int newNum){  
    this.num=newNum;  
}
```

```
public boolean isStringNull (String var){  
    return var==null;  
}
```

Conditionals

Java supports the usual logical conditions from math:

- Less than: **a < b**
- Less than or equal to: **a <= b**
- Greater than **a > b**
- Greater than or equal to **a >= b**
- Equal to **a == b**
- Not Equal to **a != b**

Java has the following conditional statements:

- If ... else
- Switch

If ... else

Complete Syntax:

```
if (<condition>) {  
    //block of code to be executed if the condition is true  
}else if(<condition>){  
    //block of code to be executed if the 2nd condition is true  
}else{  
    //block of code to be executed if any condition is false  
}
```



Examples

```
if (20 > 18) {  
    System.out.println("20 is greater than 18");  
}
```

```
int x = 20;  
int y = 18;  
if (x > y) {  
    System.out.println("x is greater than y");  
}
```

```
String name = "carol";
```

```
if (name.equals("carol")) {  
    System.out.println("Name1 is equals to Name2");  
}
```

Examples

```
if (time < 18) {  
    System.out.println("Good day.");  
} else {  
    System.out.println("Good evening.");  
}
```

```
boolean active = true;  
if(active == true){  
    System.out.println("A variable is active");  
}else{  
    System.out.println("A variable is not active");  
}
```

```
String name = "carol";
```

```
if (name.equals("carol")) {  
    System.out.println("Name is equals to carol");  
}else{  
    System.out.println("There is another name");  
}
```

Examples

```
int time = 22;  
if (time < 10) {  
    System.out.println("Good morning.");  
} else if (time < 20) {  
    System.out.println("Good day.");  
} else {  
    System.out.println("Good evening.");  
}
```

```
String name = "andrea";  
  
if (name.equals("carol")) {  
    System.out.println("Name is equals to carol");  
} else if (name.equals("andrea")) {  
    System.out.println("Name is equals to andrea");  
} else if (name.equals("catalina")) {  
    System.out.println("Name is equals to catalina");  
} else if (name.equals("ana")) {  
    System.out.println("Name is equals to ana");  
} else {  
    System.out.println("There is another name");  
}
```

If ... else Ternary Operator

Short Syntax

variable = (condition) ? expressionTrue : expressionFalse;

```
int time = 20;  
if (time < 18) {  
    System.out.println("Good day.");  
} else {  
    System.out.println("Good evening.");  
}
```



```
int time = 20;  
String result = (time < 18) ? "Good day." : "Good evening.";  
System.out.println(result);
```

Switch

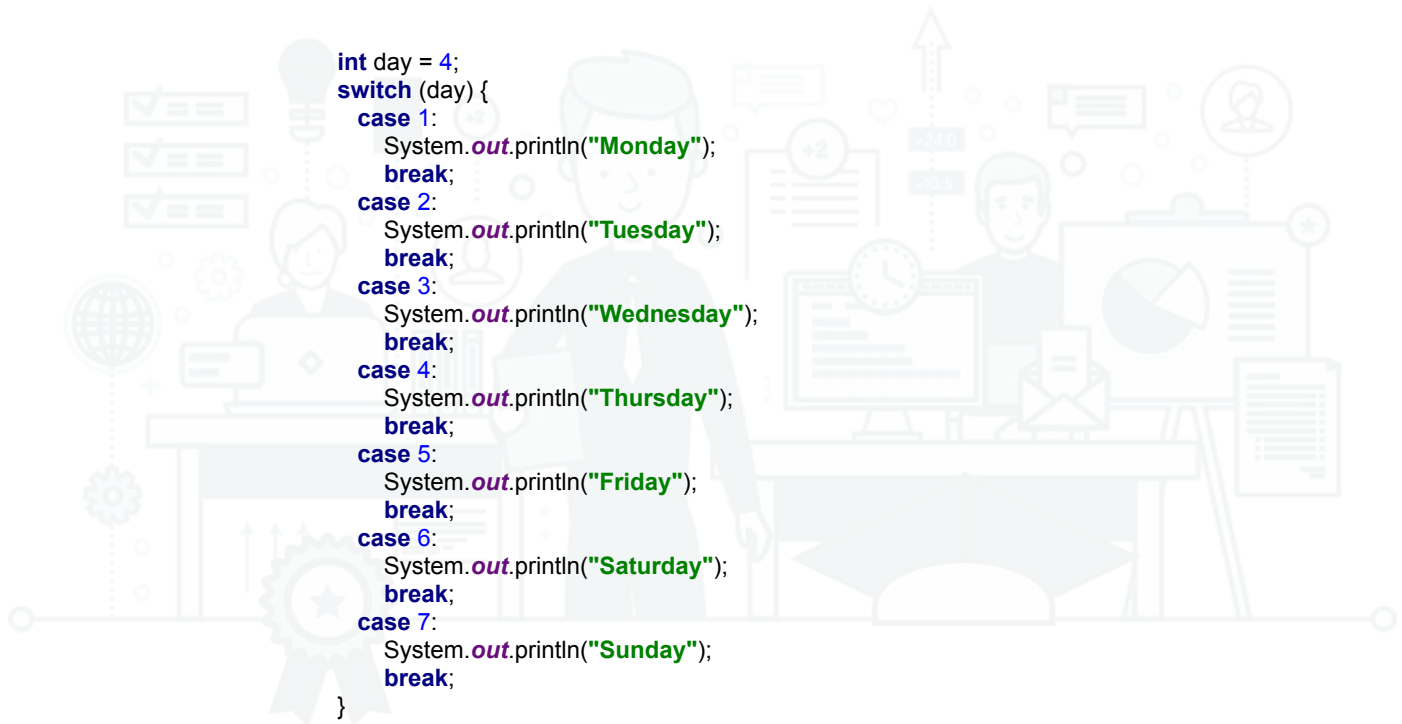
Syntax:

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
        break;  
}
```

How it works:

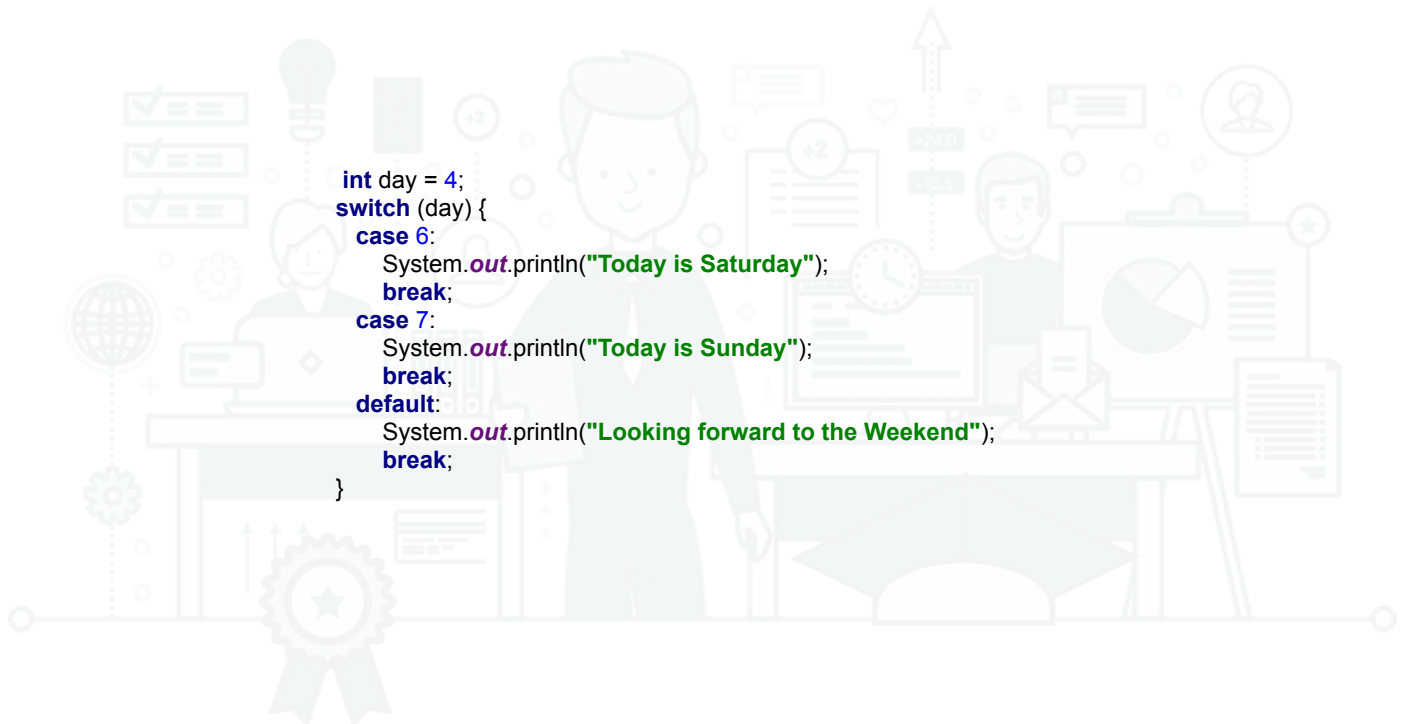
- The expression is evaluated once
- The value of the expression is compared with the values of each **case**
- If there is a match, the associated block of code is executed.
- When Java reaches a break keyword, it **breaks** out of the switch

Examples



```
int day = 4;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  case 4:
    System.out.println("Thursday");
    break;
  case 5:
    System.out.println("Friday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
  case 7:
    System.out.println("Sunday");
    break;
}
```

Examples

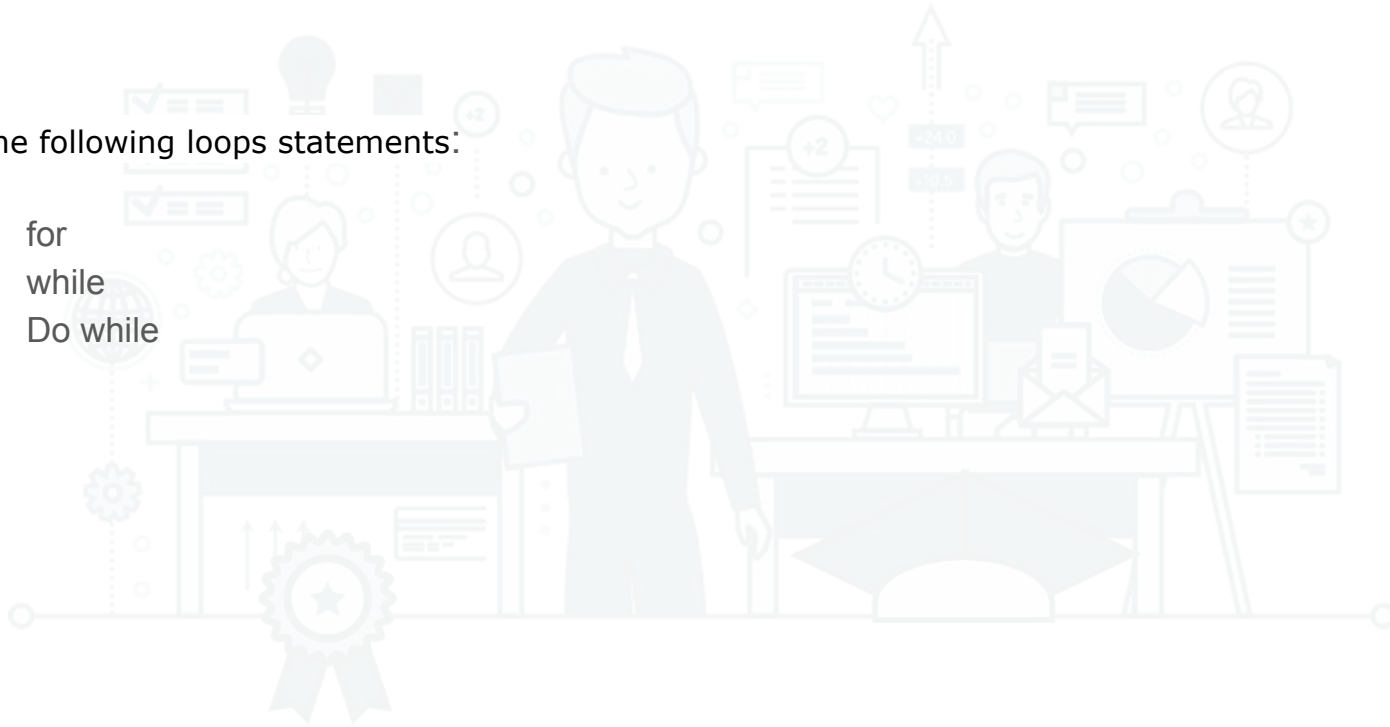


```
int day = 4;  
switch (day) {  
    case 6:  
        System.out.println("Today is Saturday");  
        break;  
    case 7:  
        System.out.println("Today is Sunday");  
        break;  
    default:  
        System.out.println("Looking forward to the Weekend");  
        break;  
}
```

Loops

Java has the following loops statements:

- for
- while
- Do while



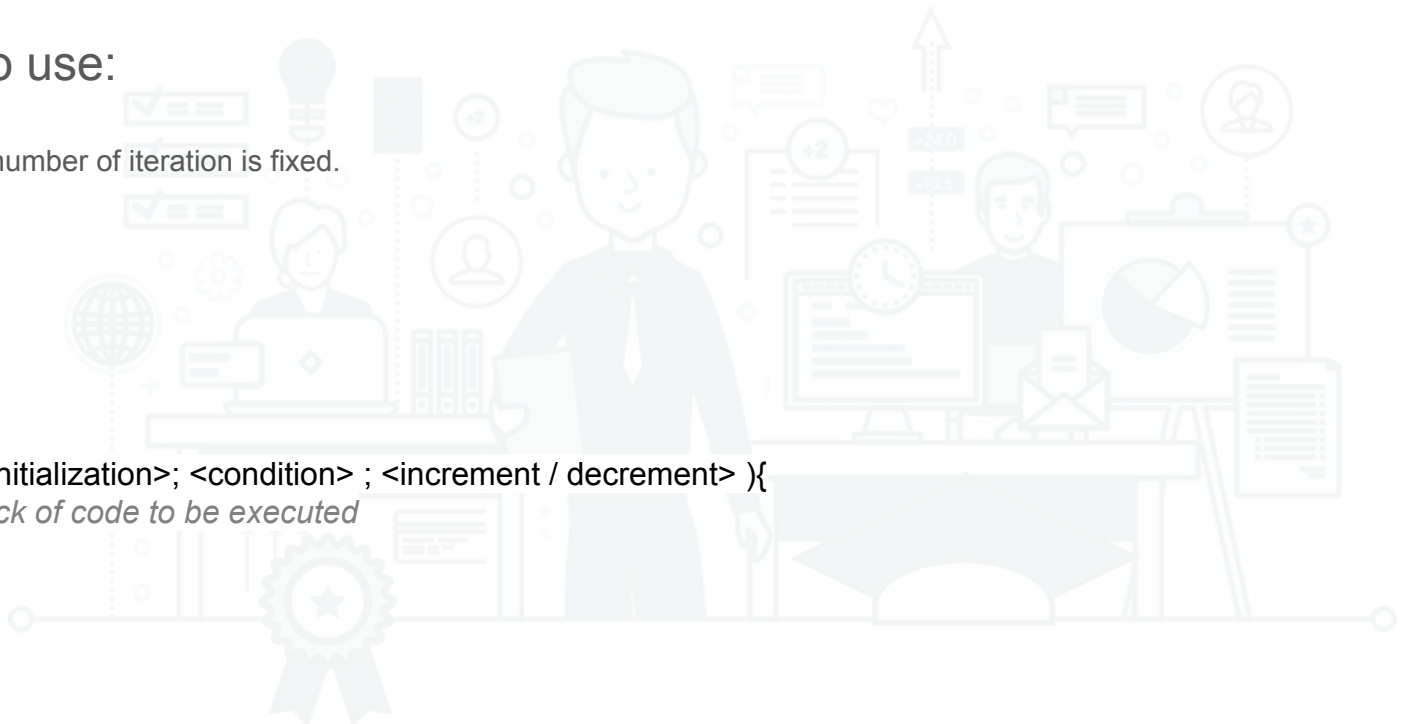
For

When to use:

If the number of iteration is fixed.

Syntax:

```
for(<initialization>; <condition> ; <increment / decrement> ){  
    //block of code to be executed  
}
```



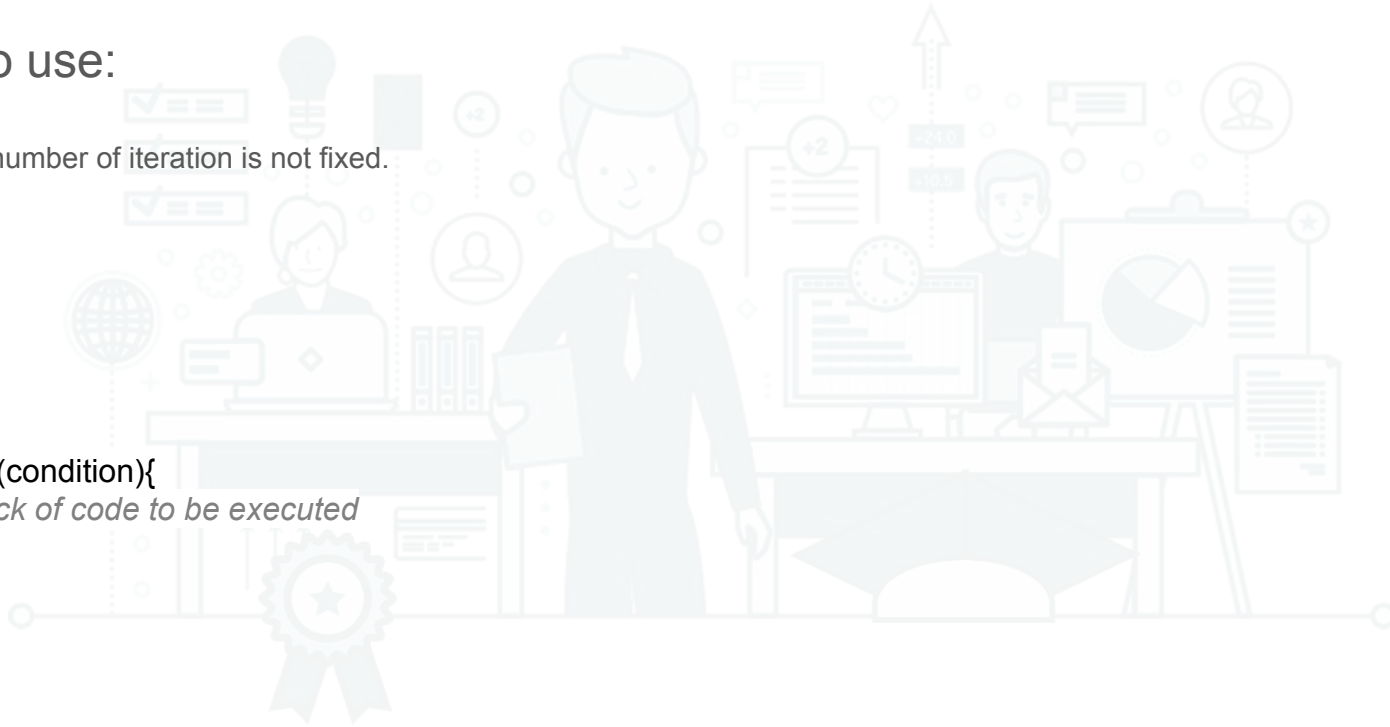
While

When to use:

If the number of iteration is not fixed.

Syntax:

```
while(condition){  
    //block of code to be executed  
}
```



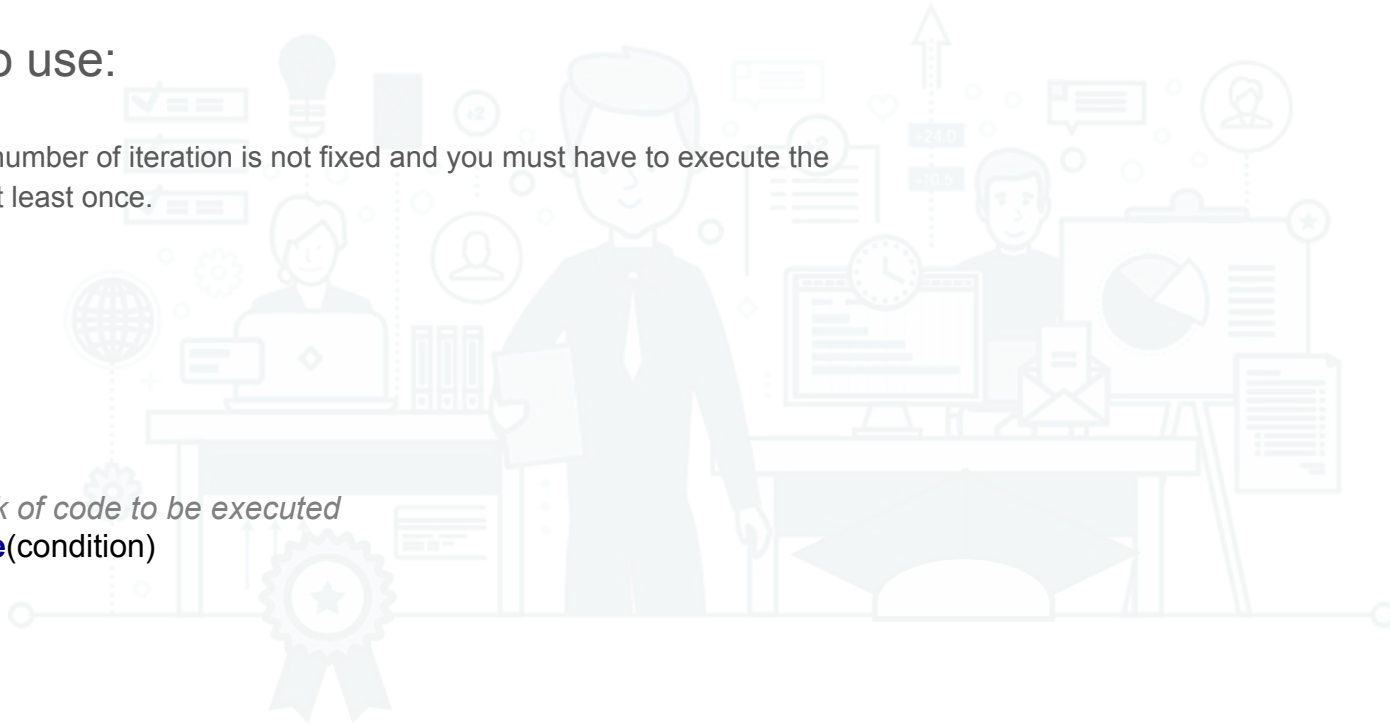
Do While

When to use:

If the number of iteration is not fixed and you must have to execute the loop at least once.

Syntax:

```
do{  
  //block of code to be executed  
}while(condition)
```



Examples

FOR

```
for(int i=1;i<=10;i++){  
    System.out.println(i);  
}
```

WHILE

```
int i = 1;  
while (i <= 10) {  
    System.out.println(i);  
    i++;  
}
```

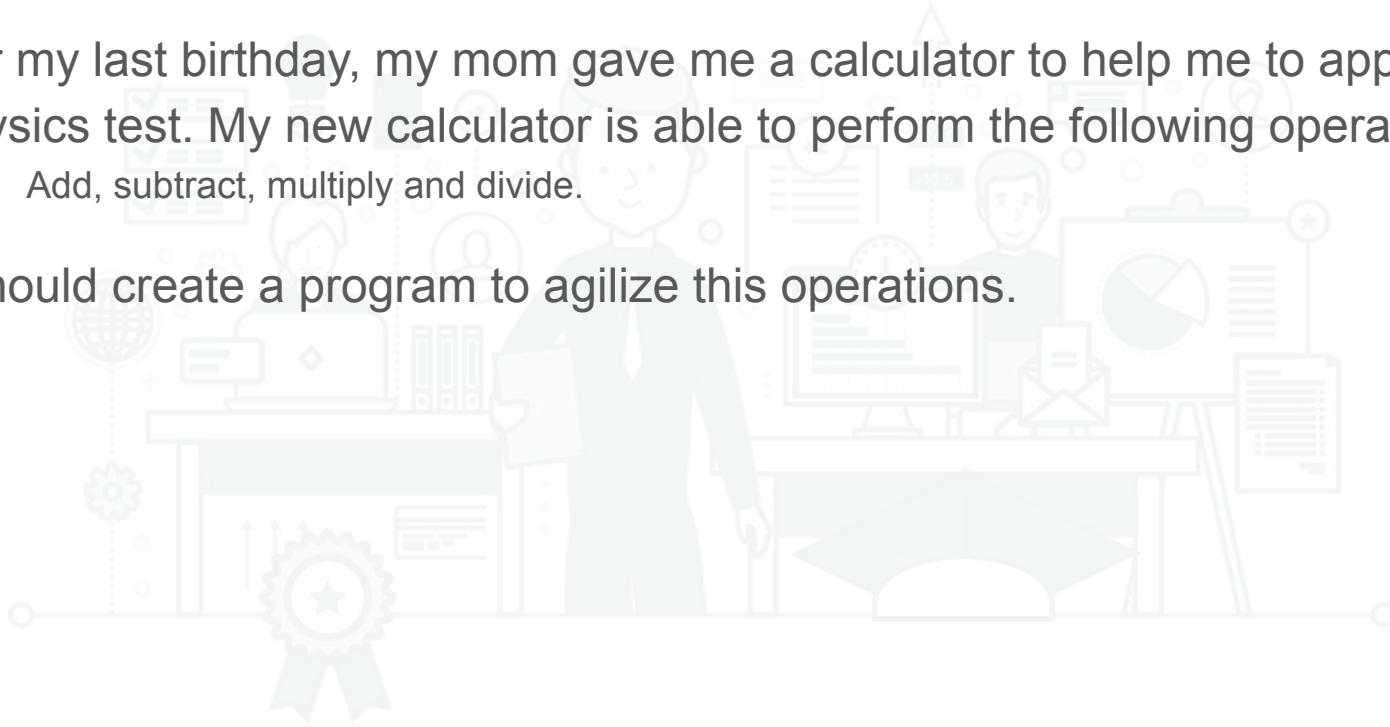
DO WHILE

```
int i = 1;  
do {  
    System.out.println(i);  
    i++;  
} while (i <= 10);
```

Exercises

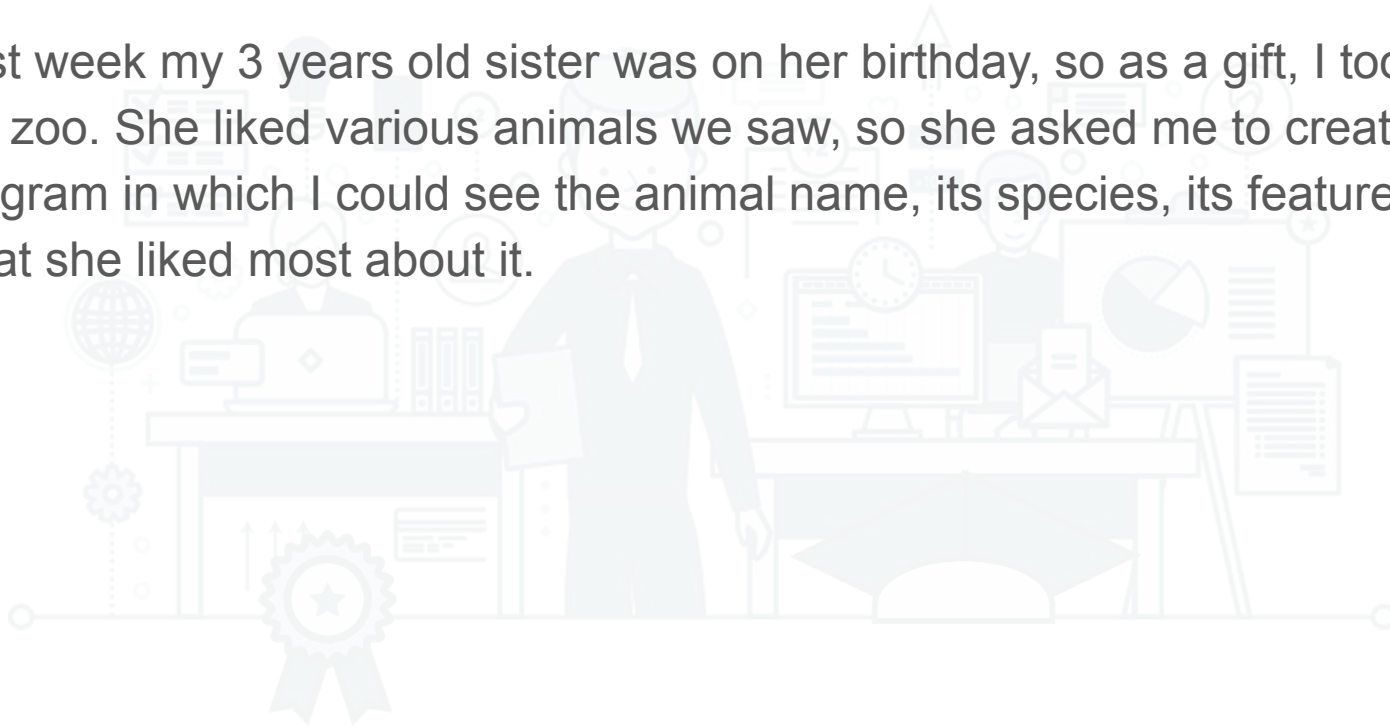
- For my last birthday, my mom gave me a calculator to help me to approve my physics test. My new calculator is able to perform the following operations:
 - Add, subtract, multiply and divide.

I should create a program to agilize this operations.



Exercises

- Last week my 3 years old sister was on her birthday, so as a gift, I took her to the zoo. She liked various animals we saw, so she asked me to create a program in which I could see the animal name, its species, its features and what she liked most about it.



Exercises

- In order to become my own boss, I gonna start a supermarket business. I need a program to help me:
 - Add the new products I get on my inventory to sell in the supermarket.
 - List the selling prices of every product
 - Sell the available products
 - Remove from my inventory the sold products

