Globant

Java and OOP
Basics

Quality Engineering Studio

Bogotá - Colombia

Agenda

- Methods
- Static modifier
- Collections

Globant

Methods (Again!)

Accessor methods: Provide access to the Account class's attributes.

Bad practice

```
account.name = "Barry Burd";
```

In fact, it should not be allowed since attributes should be private on every class

Getters:

```
public String getName(){
    return this.name;
}
```

Usage: `person.getFirstName();`

Setters:

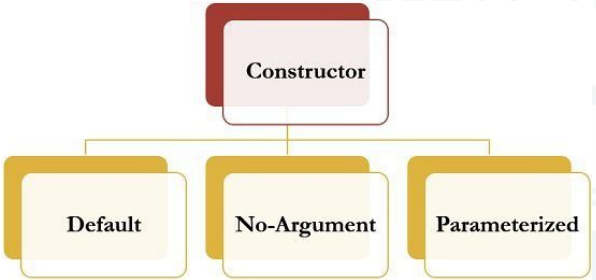
```
public void setName(String name){
    this.name = name;
}
```

Usage: `person.setFirstName("Jhon");`

Good news! You can generate getters and setters with the IDE

Methods (Again!)

Constructor: used to initialize the object's state. Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the class. Constructor(s) of a class must has **same name as the class** name in which it resides.

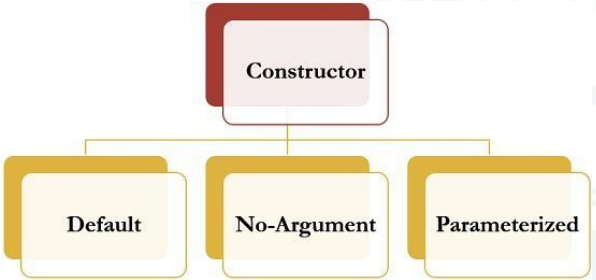



```
public Account(){
}
```

Usage: `Account myAccount = new Account();`

Methods (Again!)

Constructor: used to initialize the object's state. Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the class. Constructor(s) of a class must has **same name as the class** name in which it resides.

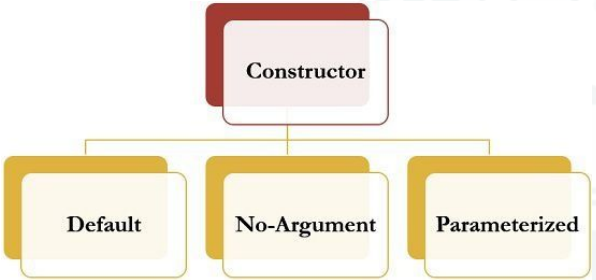



```
public Account(){
        this.name = "John Snow";
        this.address = "Dorado 45 st.";
        this.balance = 200;
}
```

Usage: `Account myAccount = new Account();`

Methods (Again!)

Constructor: used to initialize the object's state. Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the class. Constructor(s) of a class must has **same name as the class** name in which it resides.




```
public Account(String name, String address, double balance){
     this.name = name;
     this.address = address;
     this.balance = balance;
}
```

```
Usage: Account myAccount = new Account("John Snow","Dorado 45 st.", 200 );
```

Static

When the static keyword is used it implies there are class attributes or methods. In that case the element is unique for all instances (objects) of the class (it occupies a single place in memory).

On attributes:

```
<Access Modifier> static <Data type> <name>;
```

On methods:

```
<Access Modifier> static <Return Data type> <name> (<parameters..>){
      <Method body>
};
```

Example 1

```
public class Calculator{

      public int sum(int a, int b) {
         return a+b;
      }
}
```

Non Static

```
import Calculator.*;

public static void main(String[] args) {

      Calculator calculator =  new Calculator ();
      int result = calculator.sum( 20,5);
}
```

Example 1

```
public class Calculator{

      public static int sum(int a, int b) {
         return a+b;
      }
}
```

Static

```
import Calculator

public static void main(String[] args) {

      int result = Calculator.sum(20,5);

}
```

Example 2

```
public class Person{

      private String name ="Pedro";

      public String getName() {
         return name;
      }
}
```

Non Static

```
import Person.*;

public static void main(String[] args) {

      Person person1 = new Person();
      String name = person1.getName();
}
```

Example 2

```
public class Person{

      private static String name ="Pedro";

      public static String getName() {
         return name;
      }
}
```

Static

```
import static Person.*;

public static void main(String[] args) {

      String name = getName();
}
```

Collections

Data structures to organize in different ways data groups of any type or class.

Example: String

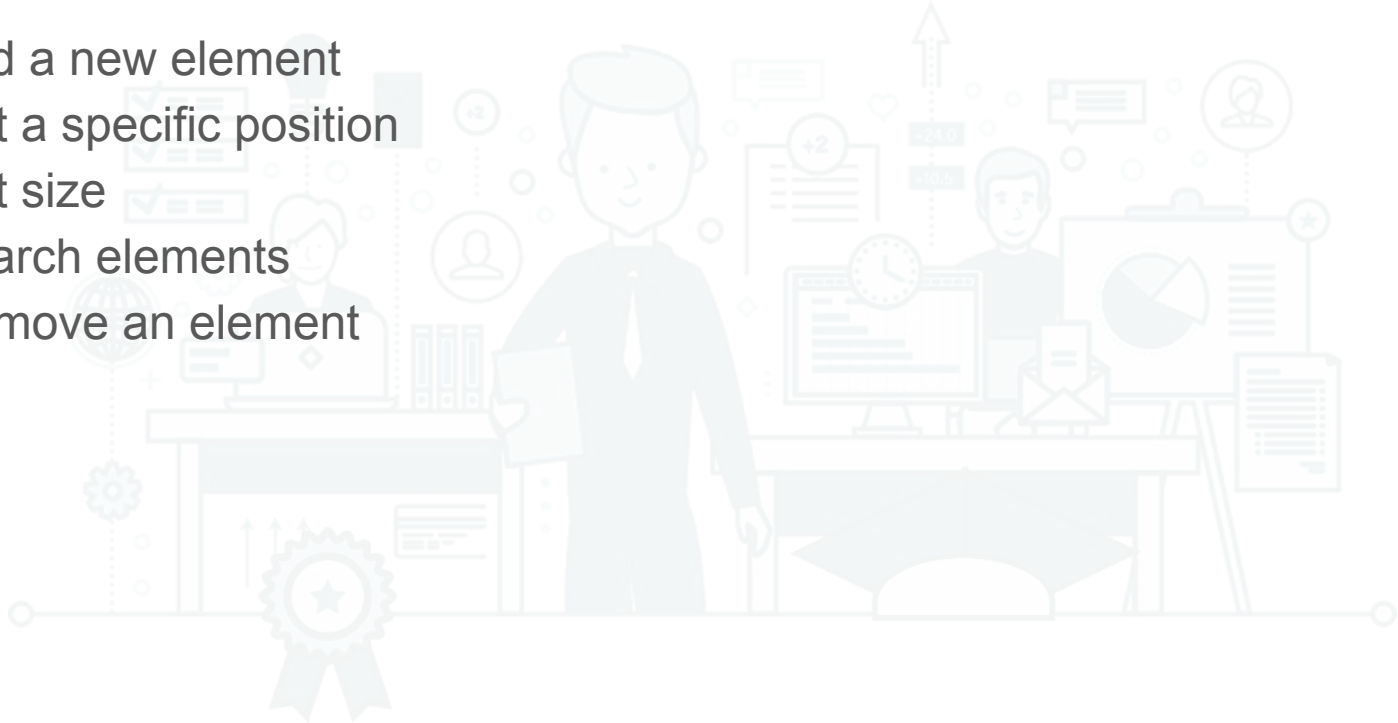
Collections

Data structures to organize in different ways data groups of any type or class. Some collection types on java are:

- *List (Data list accessible on any point)*
- Set (Unordered group with non repeated data)
- Map (Data group with structure value - key)
- Queue (First In First Out)
- Dequeue (Double Ended Queue)

Collections: Operation

- Add a new element
- Get a specific position
- Get size
- Search elements
- Remove an element



List

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

      List<Integer> myList= new ArrayList<Integer>();
      myList.add(3);
      myList.add(4);
      myList.add(6);

      myList.set(1,5);

      for (int i=0; i<myList.size(); i++){
         System.out.println(myList.get(i));
      }
}
```

Example

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

      ArrayList<String> names = new ArrayList<>();

      names.add("Felipe");
      names.add("Carlos");
      names.add("María");

      names.get(0);
      int size = names.size();
      names.remove(1);
      int size2 = names.size();
      int index= names.indexOf("María");
      names.get(index);
}
```

Exercise

- In order to become my own boss, I'm gonna start a supermarket business and I need a program that helps me to:
  - 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

Homework

I have a restaurant, with its respective name and menú. Each option on the menú contains its name and price. Make a program to:

1. Add 5 recipes to the menu
2. Replace the third option for a vegan recipe
3. Print the amount of recipes on the menu
4. Print the whole menu specifying name and price.