Constructors

Constructors

1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Default constructor available

1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Using default constructor

```
Car coche = new Car();
```

1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Using default constructor

```
Car coche = new Car();
```

1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Using default constructor

```
Car coche = new Car();
```

1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Using default constructor

```
Car coche = new Car();
```

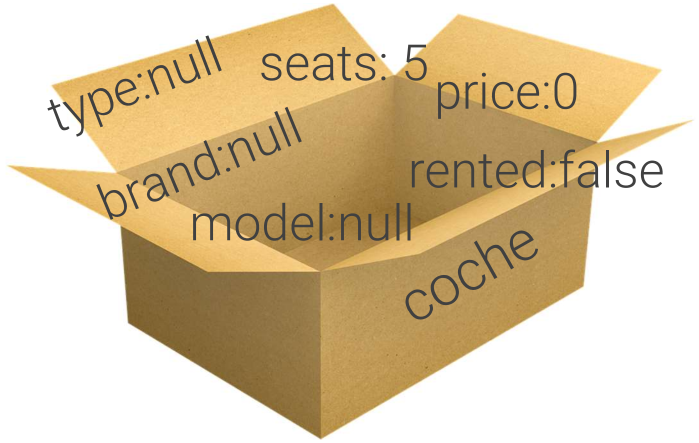
1. Constructors

```
public class Car{

  String type;
  String brand;
  String model;
  int seats=5;
  double price;
  boolean rented;

}
```

Attributes declaration

Using default constructor

```
Car coche = new Car();
```



1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

Constructor **without** input parameters

Constructor **with** input parameters

1. Constructors

```
public class ClassName{

  //Attributes



  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

Constructors:

- Have the same name as the public class itself

1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

Constructors:

- Have the same name as the public class itself
- Don’t have any return types

1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

After attributes declaration

Before methods declaration

1. Constructors

```
public class ClassName{

  //Attributes


  public ClassName(){
    //constructor block of code
    //Attributes initialization
  }

  public ClassName(parameters){
    //constructor block of code
    //Attributes initialization
  }

  //Methods

}
```

Constructor block of code:

- Attributes initialization

1. Constructors

Renting a car

```
public class Car{

   String type;
   String brand;
   String model;
   int seats;
   double price;
   boolean rented;

}
```

Car

type: Berlina
brand: Ford
model: Ka
seats: 4
price: 16.99 €/dia
~~rented: false~~

Car

type: Monovolumen
brand: Opel
model: Zafira
seats: 5
price: 26.99 €/dia
~~rented: false~~

1. Constructors

Renting a car

```
public class Car{

  String type;
  String brand;
  String model;
  int seats;
  double price;
  boolean rented;

}
```

Car

type: Berlina
brand: Ford
model: Ka
seats: 4
price: 16.99 €/dia
~~rented: false~~

Car constructor input parameters

1. Constructors

Renting a car

```
public class Car {
   String type;
   String brand;
   String model;
   int seats;
   double price;
   boolean rented;

    public Car(String t, String b, String m, int s, double p) {
        type = t;
        brand = b;
        model = m;
        seats = s;
        price = p;
        rented = false;
    }
}
```

1. Constructors

Renting a car

```
public class Car {
   String type;
    String brand;
    String model;
    int seats;
    double price;
    boolean rented;

    public Car(String t, String b, String m, int s, double p) {
        type = ?;
        brand = ?;
        model = ?;
        seats = ?;
        price = ?;
        rented = ?;
    }
}
```

Constructor block of code:
Attributes initialization

1. Constructors

Renting a car

```
public class Car {
   String type;
   String brand;
   String model;
   int seats;
   double price;
   boolean rented;

   public Car(String t, String b, String m, int s, double p) {
      type = t;
      brand = b;
      model = m;
      seats = s;
      price = p;
      rented = false;
   }
}
```

Constructor block of code:
Attributes initialization

1. Constructors

Renting a car

```
public class Car {
  String type;
  String brand;
  String model;
  int seats;
  double price;
  boolean rented;

  public Car(String t, String b, String m, int s, double p) {
     type = t;
     brand = b;
     model = m;
     seats = s;
     price = p;
     rented = false;
  }
}
```

1. Constructors

A Game

```
public class Game{

   int score;

    public Game() {
        score=0;
    }
    public Game(int startingScore){
        score=startingScore;
    }
}
```

Constructor **without** input parameters

Constructor **with** input parameters

1. Constructors

A Game

```
public class Game{

   int score;

    public Game() {
       score=0;
    }
    public Game(int startingScore){
       score=startingScore;
    }
}
```

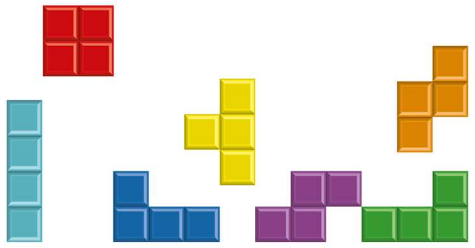
Multiple constructors

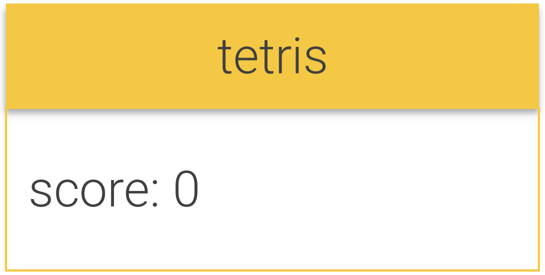
Acessing a constructor

2. Accesing a constructor

A Game

```
Game tetris= new Game();
```





2. Accesing a constructor

A Game

```
Game darts= new Game(500);
```

darts

score: 500

2. Accesing a constructor

A Game

```
Game darts= new Game(500);
System.out.println(darts.score);
```

darts

score: 500

2. Accesing a constructor

A Game

```
Game darts; //darts is null
```

2. Accesing a constructor

A Game

```
Game darts; //darts is null
System.out.println(~~darts.score);~~
```

NullPointerException

2. Accesing a constructor

Renting a car

```
Car opelZafira= new Car(”Monovolumen", ”Opel", ”Zafira", 5, 26.99);
```

opelZafira

type: Monovolumen
brand: Opel
model: Zafira
seats: 5
price: 26.99 €/dia
rented: false

Default vs defined constructors

3. Default vs defined constructors

Renting a car

```
public class Car {
   String type;
   String brand;
   String model;
   int seats;
   double price;
   boolean rented;

}
```

Default constructor available

```
Car myCar= new Car();
```

3. Default vs defined constructors

Renting a car

```
public class Car {
   String type;
   String brand;
   String model;
   int seats;
   double price;
   boolean rented;

   public Car(String t, String b, String m, int s, double p) {
      type = t;
      brand = b;
      model = m;
      seats = s;
      price = p;
      rented = false;
   }
}
```

No default constructor available

3. Default vs defined constructors

Renting a car

```
Car opelZafira= new Car(”Monovolumen", ”Opel", ”Zafira", 5, 26.99);
Car myCar= ~~new Car();~~
```

Error compilation

3. Default vs defined constructors

Renting a car

```
public class Car {
   String type;
    String brand;
    String model;
    int seats;
    double price;
    boolean rented;

    public Car(){
       seats = 5;
       rented = false;
    }
    public Car(String t, String b, String m, int s, double p) {
       type = t;
       brand = b;
       model = m;
       seats = s;
       price = p;
       rented = false;
    }
}
```

Default constructor available

3. Default vs defined constructors

Renting a car

```
Car opelZafira= new Car(”Monovolumen", ”Opel", ”Zafira", 5, 26.99);
Car miCoche= new Car();
```

opelZafira	miCoche
type: Monovolumen	type: null
brand: Opel	brand: null
model: Zafira	model: null
seats: 5	seats: 5
price: 26.99 €/dia	price: 0 €/dia
rented: false	rented: false

3. Default vs defined constructors

A Game

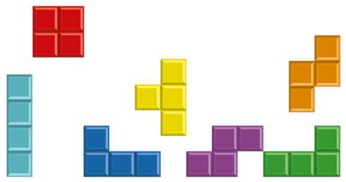
```
public class Game{

   int score;

    public Game() {
       score=0;
    }
    public Game(int startingScore){
       score=startingScore;
    }
}
```

3. Default vs defined constructors

A Game

Game tetris= **new** Game();

Game darts= **new** Game(500);



tetris

score: 0

darts

score: 500

This keyword

4. This keyword

```
public class Car {

    String type;
    String brand;
    String model;
    int seats;
    double price;
    boolean rented;

    public Car(String t, String b, String m, int s, double p) {
        type = t;
        brand = b;
        model = m;
        seats = s;
        price = p;
        rented = false;
    }
}
```

Constructor block of code

4. This keyword

```
public class Car {

    String type;
    String brand;
    String model;
    int seats;
    double price;
    boolean rented;

    public Car(String type, String brand, String model, int seats, double price) {
        this.type = type;
        this.brand = brand;
        this.model = model;
        this.seats = seats;
        this.price = price;
        this.rented = false;
    }
}
```

4. This keyword

```
public class Car {

    String type;
    String brand;
    String model;
    int seats;
    double price;
    boolean rented;

    public Car(String type, String brand, String model, int seats, double price) {
        this.type = type;
        this.brand = brand;
        this.model = model;
        this.seats = seats;
        this.price = price;
        this.rented = false;
    }
}
```

this keyword

“No juzgues cada día por la cosecha que recoges, sino por las semillas que plantas.“

Robert Louise Stevenson, novelista y poeta escocés

La Salle