Methods

1. Method implementations
2. Accessing object methods
3. Overloading methods

Method implementations

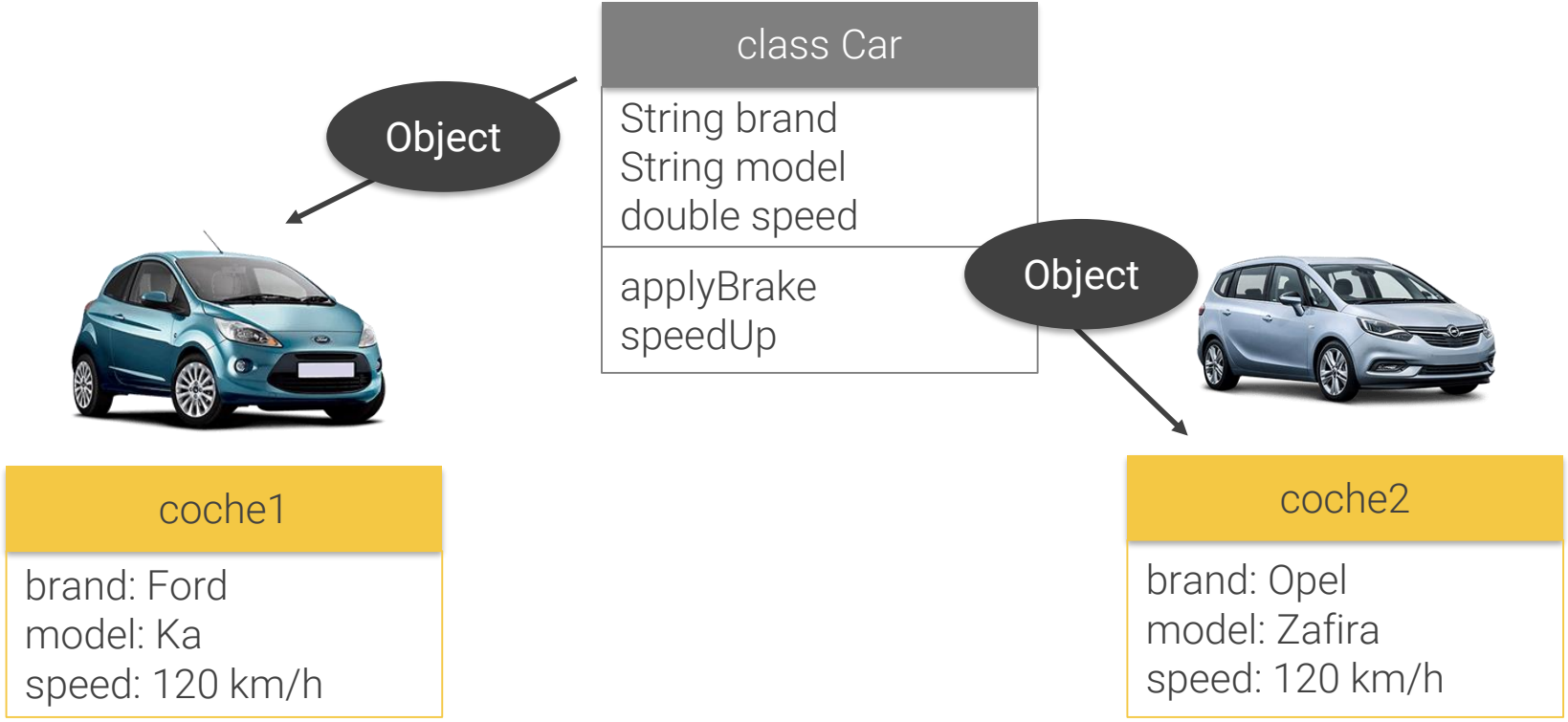
1. Method implementations

```
public class MyClass {

 // field declarations        1
 // constructors              2
 // method implementations    3

}
```

1. Method implementations

Métodos (methods): acciones que podemos realizar sobre un objeto

1. Method implementations



1. Method implementations

Métodos (methods): acciones que podemos realizar sobre un objeto

Dependen del escenario

1. Method implementations

APPLICATION: CAR RACING GAME

class Car
String brand String model double speed
applyBrake speedUp

APPLICATION: RENT A CAR

class Car
String type String brand String model int seats double price boolean isRented
rentPrice setRented

1. Method implementations

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

//constructors

//method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
       return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

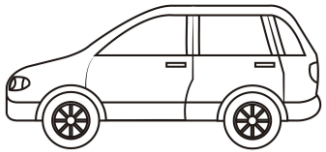
Access modifier

Method name
(lower Camel Case Rule)

Parameters
(parameter1, parameter2, ...)

Method implementations

Return type or Void
on declaration



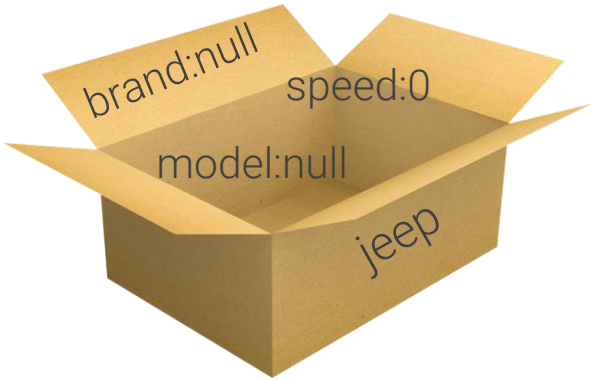
Accessing object methods

2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

```
Car jeep = new Car();
jeep.speedUp(100);
```

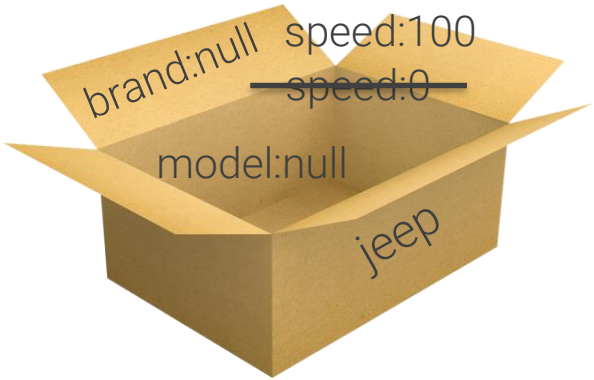
Create an object



2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

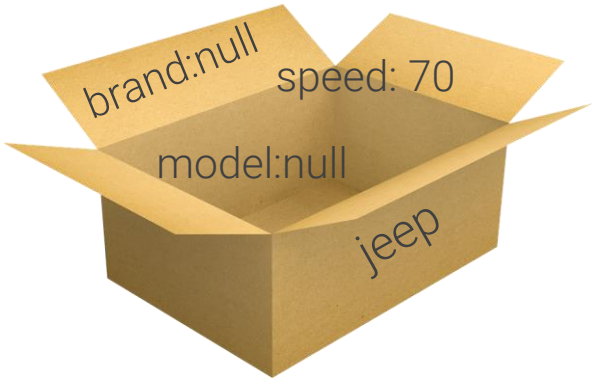
```
Car jeep = new Car();
jeep.speedUp(100);
```



2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

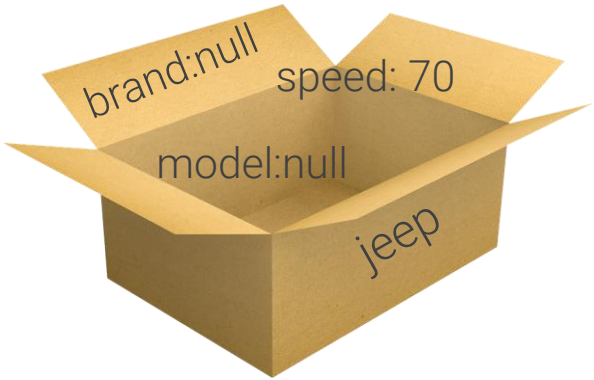
```
Car jeep = new Car();
jeep.speedUp(100);
jeep.speedUp(20);
jeep.applyBrake(50);
```



2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

```
Car jeep = new Car();
double currentSpeed = jeep.speedUp(100);
currentSpeed = jeep.speedUp(20);
currentSpeed = jeep.applyBrake(50);
System.out.println(currentSpeed);
```



2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

```
Car jeep; //jeep is null
```

2. Accessing object methods

```
public class Car{

 //field declarations
   String brand;
   String model;
   double speed;

 //constructors
 //method implementations
   public double applyBrake(double decrement) {
      speed -= decrement;
      return speed;
   }
   public double speedUp(double increment) {
      speed += increment;
      return speed;
   }
}
```

```
Car jeep; //jeep is null
double currentSpeed = jeep.speedUp(100);
```

Compilation error:
Variable 'jeep' might not have been initializated

Overloading methods

3. Overloading methods

```
public class Artist {
 ...
 public void draw(String s) { ... }
 public void draw(int i) { ... }
 public void draw(double f) { ... }

 }
```

Overloaded methods are differentiated by the number and the type of the parameters

3. Overloading methods

```
public class Artist {
 ...
 public void draw(String s) { ... }
 public boolean draw(String s) { ... }
 public void draw(int i) { ... }
 public void draw(double f) {

 }
```

You cannot declare more than one method with the same name and the same number and type of arguments

“Una persona ganadora, es una persona soñadora que nunca se rinde.“

Nelson Mandela, activista, abogado y político sudafricano conocido principalmente por luchar pacíficamente contra la segregación racial en Sudáfrica