# Class Design

## Creating Constructors

# Constructors

- special methods which are called every time you create an instance of an object

```
public class Dog {
  public Dog() { System.out.println("Woof!"); }
}
```

- name of the constructor must match the name of the class
- constructors don't have a return type (!!)
  - if the return type is stated, than it's just a normal method

```
public class Dog {
   public Dog() {
      System.out.println("Woof!");
   }
}


// in the main method
Dog dog = new Dog();
```

```
"Woof!"
```

```
// constructor overloading
public class Dog {
  private String name;
  private int age;
  public Dog() { System.out.println("Woof!"); }
  public Dog(String name, int age) {
    this.name = name;
    this.age = age;
  }
  public Dog(String name) { this.name = name; }
  public Dog(int age) { this.age = age; }
  public Dog(boolean isPuppy, String name) {
    this.age = isPuppy ? 0 : -1;
    this.name = name;
  }
  // main method...
```

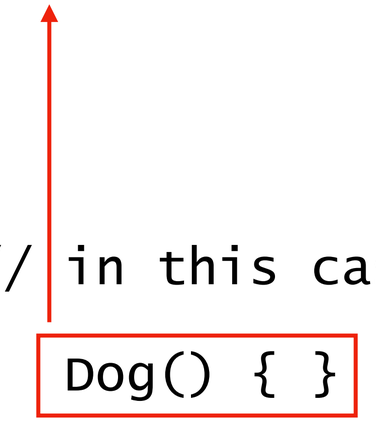
no-argument constructor

```java
// main method
public static void main(String[] args) {
    Dog dog = new Dog("Rex", 5);
    Dog puppy = new Dog(true, "Roy");
    System.out.println("Name: " + dog.name + ", Age: " + dog.age);
    System.out.println("Name: " + puppy.name + ", Age: " + puppy.age);
}
```

```
Name: Rex, Age: 5
Name: Roy, Age: 0
```

```java
public class Dog {
    private String name;
    private int age;

    public Dog(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public Dog(boolean isPuppy, String name) {
        this.age = isPuppy ? 0 : -1;
        this.name = name;
    }
```

```
// if the class has no defined constructors the default constructor is created
class Dog {
  public String name;
  public String age;
}


// in this case compiler will create no-argument constructor:
  Dog() { }


// that constructor will be called when you make an instance of the class
Dog dog = new Dog();
```



```
// default constructor is created only if no other constructor is present
class Dog {
  public String name;
  public String age;
  public Dog (String name, int age) {
    this.name = name;
    this.age = age;
  }
}
public class MyClass {
  public static void main(String[] args) {
    Dog dog = new Dog();
  }
}
```

no-argument constructor will not be auto-generated !

no-argument constructor not found => code does not compile

# Constructor access modifiers

- constructors are usually made `public`
  - but you can also make them `protected`, default or `private`
- private constructors are used if you don't want public no-argument constructor to be generated by the compiler
- in this case, the instance is usually created via some static method, and not using the keyword `new`
- we have seen this behavior in classes used to create Dates and Times, e.g.

```
LocalDate now = LocalDate.now();
```