Object Superclass

Mr. Poole
Java

Objects in Java

Reminder: Java is an **Object-Oriented Language**

Everything in Java is an **Object**

Strings are **objects**

integers are **objects**

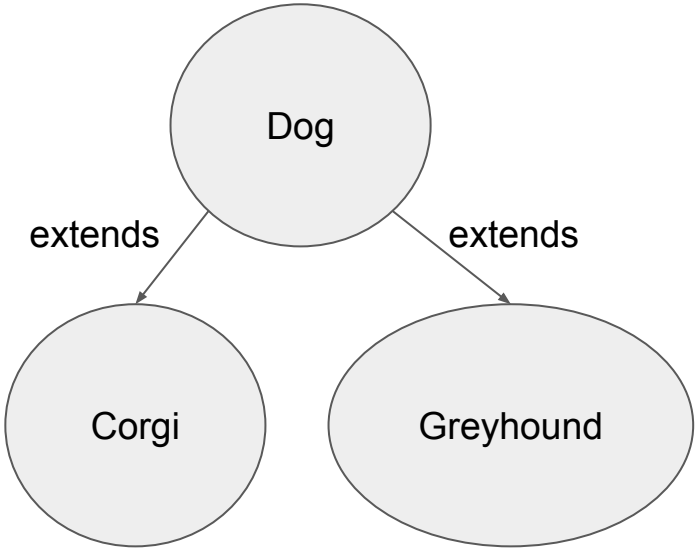
Classes are **objects**

Classes with Inheritance

So far this has been the structure of our classes.

Dog, Corgi, and Greyhound are all **Objects**

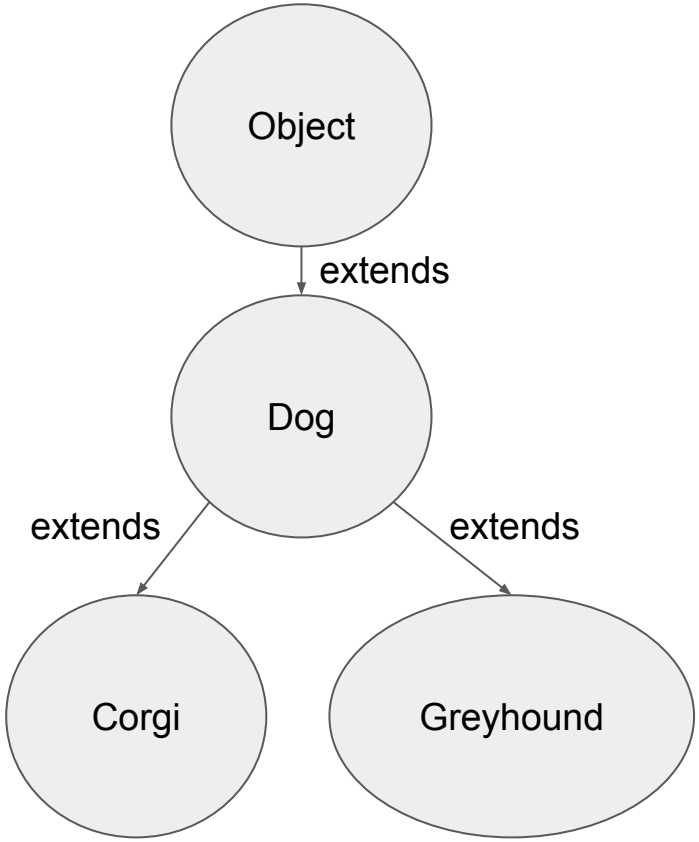
<u>But what's above Dog?!?!</u>



Object Superclass

Above Dog is the

Object Superclass



Object is the superclass to everything.

Object has two important **inherited methods**:

1. toString()
2. equals()

Let's start with **toString()**

What happens if we **print out a Dog?**

```java
public static void main(String[] args){
    Dog myobj = new Dog();
    System.out.print(myobj);
}
```

What prints here?

```java
package pkg;

public class Dog{
    String name;

    public Dog(){
        name = "Doggo";
    }
}
```

Output of Dog object

```
public static void main(String[] args){
    Dog myobj = new Dog();
    System.out.print(myobj);
}
```

This prints out the following:

```
pkg.Dog@5e91993f
```

This is the hashcode of the object, aka an ID.

Output of Dog object

```
public static void main(String[] args){
    Dog myobj = new Dog();
    System.out.print(myobj);
}
```

This prints out the following:

```
pkg.Dog@5e91993f
```

Printing objects invokes the **toString()** method from Object.

toString() allows objects to be printed but prints the hashcode seen here.

Let's **override** the **toString()** method!

toString()

- Has no parameters
- Returns a String

Here we override the toString() from Object.

```
public class Dog{
    String name;

    public Dog(){
        name = "Doggo";
    }

    public String toString(){
        return ("This is the dog, " + name);
    }
}
```

Make sure your method line looks **EXACTLY** the same!

Now when we run the following we get: `This is the dog, Doggo`

```
public static void main(String[] args){
    Dog myobj = new Dog();
    System.out.print(myobj);
}
```

Next **equals()**

What is the output of the following?

```
public static void main(String[] args){
    Dog myobj = new Dog("Bob");
    Dog myobj2 = new Dog("Bob");
    System.out.print(myobj.equals(myobj2));
}
```

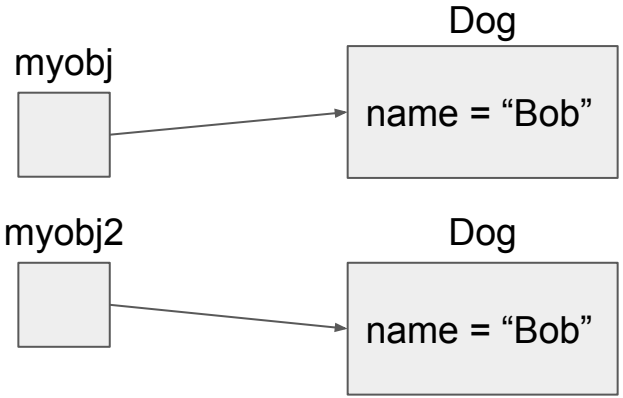
Output of **.equals()**

```
public static void main(String[] args){
    Dog myobj = new Dog("Bob");
    Dog myobj2 = new Dog("Bob");
    System.out.print(myobj.equals(myobj2));
}
```

This prints: `false`

Object equals() = False

```
public static void main(String[] args){
    Dog myobj = new Dog("Bob");
    Dog myobj2 = new Dog("Bob");
    System.out.print(myobj.equals(myobj2));
}
```

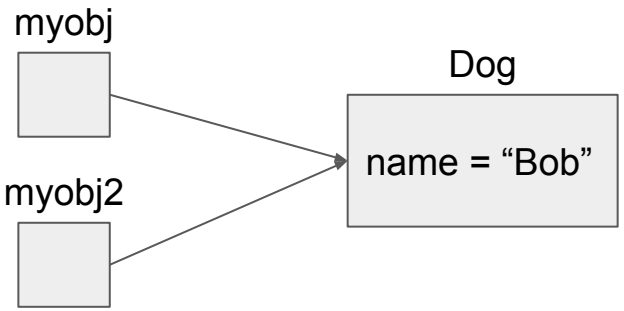


.equals() compares if the objects are the same.

In this case, we create 2 separate objects so they aren’t the same.

Object equals() = True

```
public static void main(String[] args){
    Dog myobj = new Dog("Bob");
    Dog myobj2 = myobj;
    System.out.print(myobj.equals(myobj2));
}
```



Now we changed our code to reference the same object.

This prints: `true`

Let's **override** the **equals()** method!

equals()

- Parameter Object
- Returns a boolean

We can change what we're comparing in our **overriding equals() method**

```
public class Dog{
    public String name;

    public Dog(){
        name = "Doggo";
    }
    public boolean equals(Object other){
        return this.name == other.name;
    }
}
```

Make sure your method line looks **EXACTLY** the same!

Now this **compares the names** instead of the objects! It's **true**!

```
public static void main(String[] args){
    Dog myobj = new Dog("Bob");
    Dog myobj2 = new Dog("Bob");
    System.out.print(myobj.equals(myobj2));
}
```

Now we must clarify our boolean equals() method.

Since Object is a superclass,
Object could represent any class type.

Object could be an integer, double,
String, or a Dog.

We must make sure it's a Dog so that
we can make the correct comparison.

```
public class Dog{
    public String name;

    public Dog(){
        name = "Doggo";
    }
    public boolean equals(Object other){
        return this.name == other.name;
    }
}
```

This could be anything!

Checking the Object other

This if statement was added!

This checks if the Object other is a Dog!

```
public boolean equals(Object other){
    if(!(other instanceof Dog)){
        return false;
    }
    else{
        return this.name == ((Dog)other).name;
    }
}
```

Object Casting to make sure other has a name

This casting was added!

Cast as a Dog because other doesn't have the name variable!

```
public boolean equals(Object other){
    if(!(other instanceof Dog)){
        return false;
    }
    else{
        return this.name == ((Dog)other).name;
    }
}
```

Name is a string so use .equals!

.equals instead of ==

```
public boolean equals(Object other){
    if(!(other instanceof Dog)){
        return false;
    }
    else{
        return this.name.equals(((Dog)other).name;);
    }
}
```

Summary:
Object Superclass

Two Methods

- **toString()** = Prints objects
- **equals()** = Compares objects

Override them for your objects!

Lab: Object Superclass

1. Add to your Performer class
    a. Override toString() and equals()
        i. Print out the name and age of the Performer
        ii. Check if the names are the same
2. Add to your Musician
    a. Override toString()
        i. Print out the name and instrument
3. Add to Apprentice
    a. Override toString()
        i. Print out the name, instrument, and university
4. Main
    a. On your array of 4 Performers from last lab, use toString() on all Performers
    b. Check if two Performers are the same