Overriding Methods

Mr. Poole
Java

How do we create methods with inheritance?

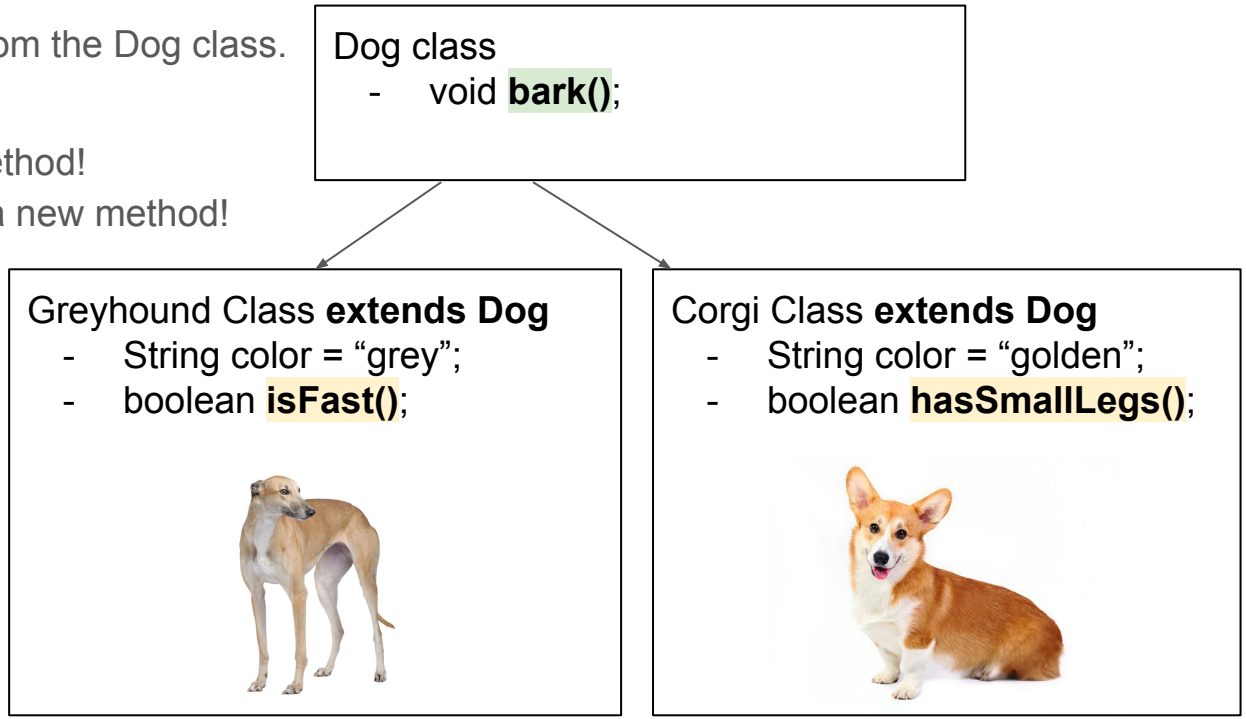
1. **Inherit methods** - Any public method in the superclass become valid public methods of the subclass. These are especially important to access private instance variables of the superclass.

2. **Write new methods** - The subclass can have additional methods that are completely independent of methods in the superclass. This includes methods that are overloaded (same method name, different signatures) and treated as independent methods.

3. **Override methods** - Write new and different implementations of a method that already exists in the super class.

Let's look back at our dogs!

1. Inherited methods
   a. **bark()** is inherited from the Dog class.
2. Write new methods
   a. **isFast()** is a new method!
   b. **hasSmallLegs()** is a new method!
3. Override Methods
   a. None!



Let's override **bark()** for the Greyhound!

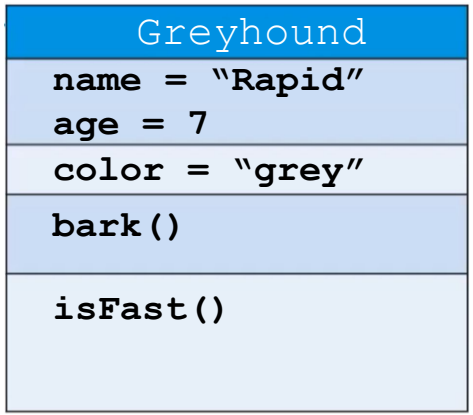
```
public class Dog{
      private String name;
      private int age;

      public Dog() {...}
      public Dog(String n, int a){...}

      public void bark(){
            System.out.println("Bark!");
      }
}
```

```
public class Greyhound extends Dog{
      private String color;

      public Greyhound () {...}
      public Greyhound (String n, int a, String c){...}

      public boolean isFast(){
            return true;
      }
}
```

Right now, this is our code, nothing has been changed.

Given the following code, what's the output?

```
Greyhound rapid = new Greyhound("Rapid", 7, "grey");
rapid.bark();
```



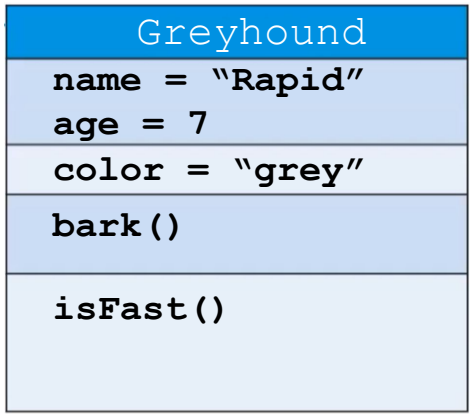
Overriding a Method

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

      public Dog() {...}
      public Dog(String n, int a){...}

      public void bark(){
            System.out.println("Bark!");
      }
}
```

This outputs: `Bark!`

```
Greyhound rapid = new Greyhound("Rapid", 7, "grey");
rapid.bark();
```

```
public class Greyhound extends Dog{
      private String color;

      public Greyhound () {...}
      public Greyhound (String n, int a, String c){...}

      public boolean isFast(){
            return true;
      }
}
```



Overriding a Method

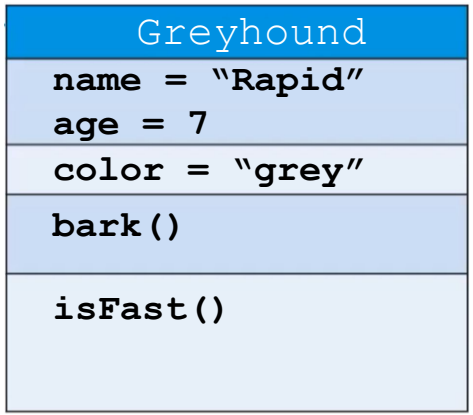
```
public class Dog{
      private String name;
      private int age;

      public Dog() {...}
      public Dog(String n, int a){...}

      public void bark(){
            System.out.println("Bark!");
      }
}
```

```
Greyhound rapid = new Greyhound("Rapid", 7, "grey");
rapid.bark();
```

So let's override **bark()**!

All we need to do, is create the same method in the Greyhound class.

```
public class Greyhound extends Dog{
      private String color;

      public Greyhound () {...}
      public Greyhound (String n, int a, String c){...}

      public boolean isFast(){
            return true;
      }
      public void bark(){
            System.out.println("LOUD BARK!");
      }
}
```



Overriding a Method

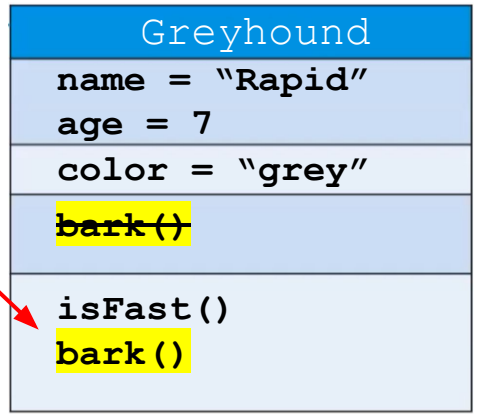
```
public class Dog{
      private String name;
      private int age;

      public Dog() {...}
      public Dog(String n, int a){...}

      public void bark(){
            System.out.println("Bark!");
      }
}
```

```
public class Greyhound extends Dog{
      private String color;

      public Greyhound () {...}
      public Greyhound (String n, int a, String c){...}

      public boolean isFast(){
            return true;
      }
      public void bark(){
            System.out.println("LOUD BARK!");
      }
}
```

```
Greyhound rapid = new Greyhound("Rapid", 7, "grey");
rapid.bark();
```

So let's override **bark()**!

Since we add bark to Greyhound, now our method **overrides** the Dog class's bark().



Overriding a Method

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

      public Dog() {...}
      public Dog(String n, int a){...}

      public void bark(){
            System.out.println("Bark!");
      }
}
```

```
public class Greyhound extends Dog{
      private String color;

      public Greyhound () {...}
      public Greyhound (String n, int a, String c){...}

      public boolean isFast(){
            return true;
      }
      public void bark(){
            System.out.println("LOUD BARK!");
      }
}
```

```
Greyhound rapid = new Greyhound("Rapid", 7, "grey");
rapid.bark();
```

Our new output for the code above is:

```
LOUD BARK!
```

Greyhound
name = "Rapid" age = 7
color = "grey"
~~bark()~~
isFast() bark()

Lab: Overriding Methods

1. Create a new class - Actor
    a. Global Variable - String type
    b. Constructors
        i. Default Constructor - type = “theater”
        ii. Name, Type Constructor
    c. Methods
        i. Override practice()
            1. Add a little spice of your own
        ii. Override perform()
            1. Add a little spice of your own
        iii. void monologue()
            1. Insert your favorite monologue
2. In main
    a. Create an actor, have it practice, perform, and monologue

Hint: in foreshadow of next lab, you can use super.getName() for these