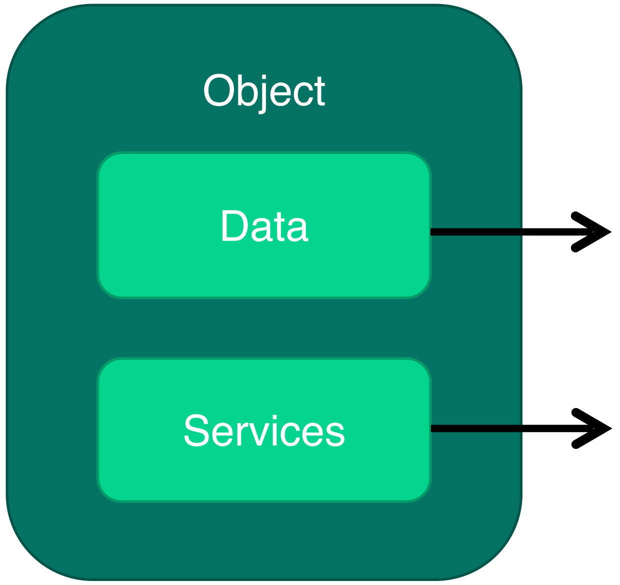
OOP – Object Oriented Programming

- In OOP we try to create a model of reality with objects
- Objects often represent things in the real world
- Objects consist of data and services (variables and methods)



```
String name;
int age;

void
doSomething(){
                ...
}
```

Objects and Object References

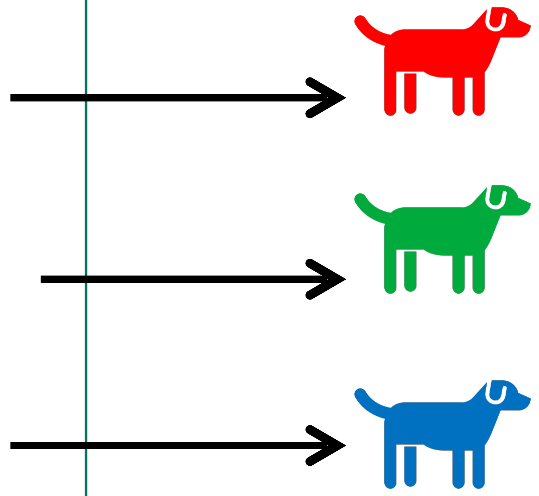
- Objects are created with the new operator
- An object is accessed through an object reference
- The object reference is a handle to an object in memory
- Variables are always object references, not objects directly
- Object references can be copied and compared, and their reference can be assigned to another object
- If an object has no references to it, it will get deleted by the Garbage Collector

Classes and Objects

- We write source code in classes
- Objects are instantiated from classes (we never write objects)
- A class is like a template to an object

```
class Dog {
    public Color color;

    public void eat() {
    }
}
```

```
public static void main(String[]
args) {
    Dog dog1 = new Dog();
    dog1.color = Color.RED;

    Dog dog2 = new Dog();
    dog2.color = Color.GREEN;

    Dog dog3 = new Dog();
    dog3.color = Color.BLUE;
}
```

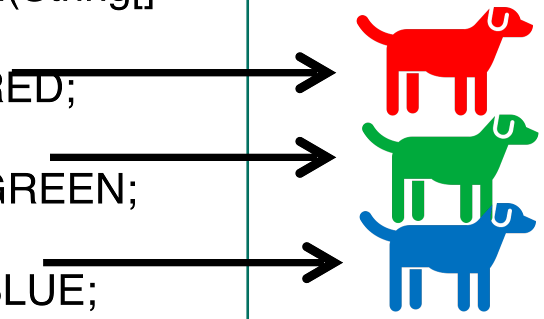


Classes and Objects

- There is only one unique class of every kind in the system
- There can be any number of unique objects instantiated from the same class in the system
- Every object has its own unique data
- Most things we will use in programs are objects
- Static variables belongs to the class, not the object

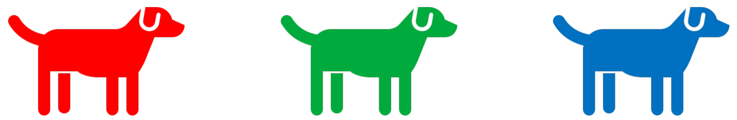
```
class Dog {
    public static Color color;

    public void eat() {
    }
}
```

```
Public static void main(String[] args) {
    Dog.color = Color.RED;

    Dog.color = Color.GREEN;

    Dog.color = Color.BLUE;
}
```



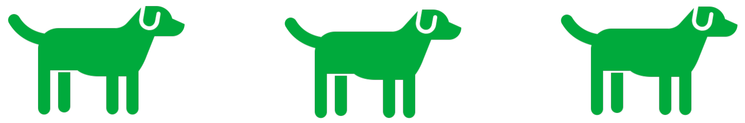
Instance variables

- Instance variables are declared in the class, not in a method
- Not being declared as static makes a variable non-static
- Non-static variables belong to the instance and not to the class
- Non-static variables are not accessible from a static method
- If the Dog class has a non-static color variable it belongs to the object (instance).
- Then every unique Dog object will have it’s own unique color



Static variables

- Static variables are declared with the static keyword
- Static variables belong to the class
- Static variables are accessible from instances (!)
- The value of a static variable is the same for all instances of the class
- If the color variable in the Dog class is static, there will only be one variable for all Dog objects, setting it to one color makes all dogs have the same color



Instance methods and Static methods

- Not being declared as static makes a method belong to the instance and not to the class
- A method in the class declared as static belongs to the class
- A static method can’t access non-static variables or non-static methods since these belong to an object and not the class

Wrapper Objects to primitive data types

- All primitive datatypes have a corresponding wrapper object
-
```
Integer number = new Integer(37); // example of declaring an Integer
```

Type	Wrapper Object
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Objects in arrays

- Arrays can not only have primitive types as elements
- Arrays can also have object references as elements (not objects)
- The value of an element that has not been assigned a value is null in an array of Objects

```
Dog[] dogs = new Dog[3]; // creates a Dog array

Dog dog = dogs[0]; // value of dog is null
dogs[1] = new Dog(); // assigns a dog object to index 1
```

Iteration with an array of Objects

- Same rules apply with objects, you can use a for loop:

```
for (int i = 0; i < dogs.length;i++) {
    System.out.println(dogs.[i].getName());
}
```

- Or a ”for-each” loop:

```
for (Dog dog : dogs) {
    System.out.println(dog.getName());
}
```

- Just remember, if you try to access methods on the dog object reference and it isn’t assigned an object, you will get a null pointer exception!