Java Programming I

Session 5

Classes and Objects

Juan Carlos Moreno - UCLA Ex

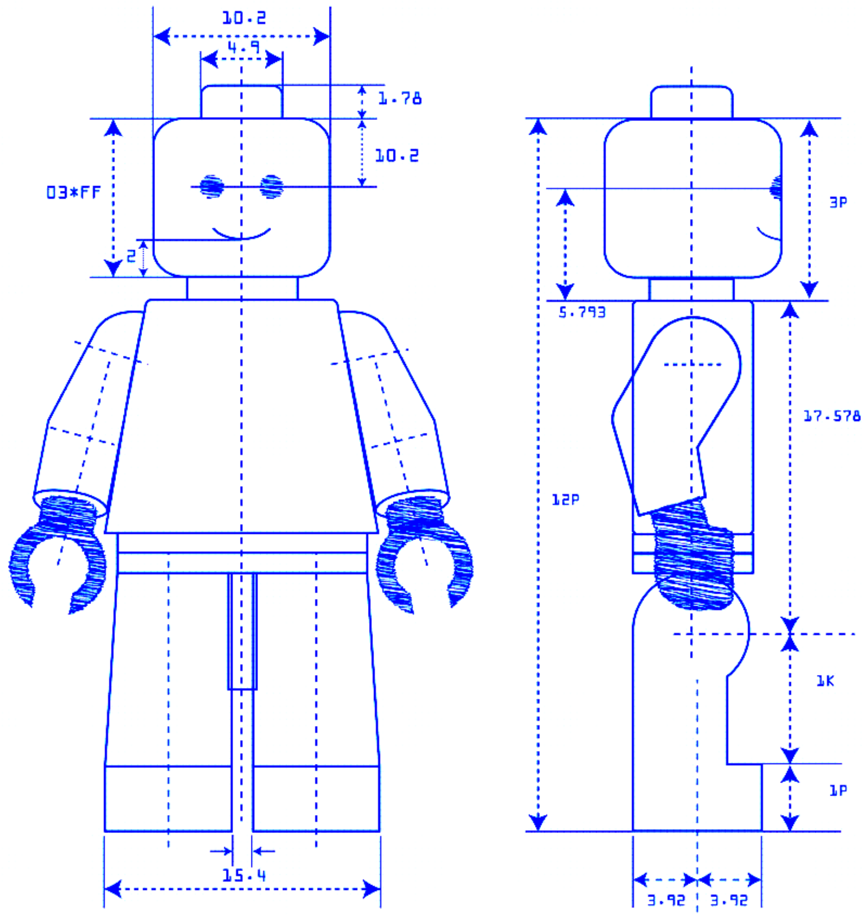
Agenda

- **Classes**
- **Objects**
- **Methods**
- **Constructors**
- **Method overloading**

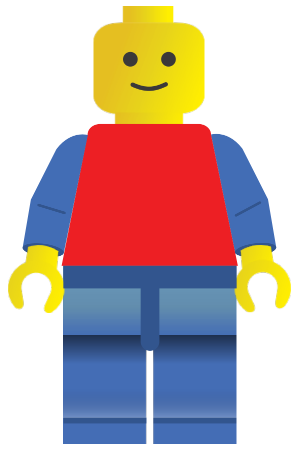
Classes

Show me all the blueprints

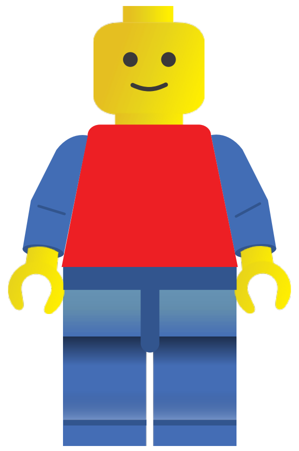
public class LegoMan



new LegoMan();



new LegoMan();

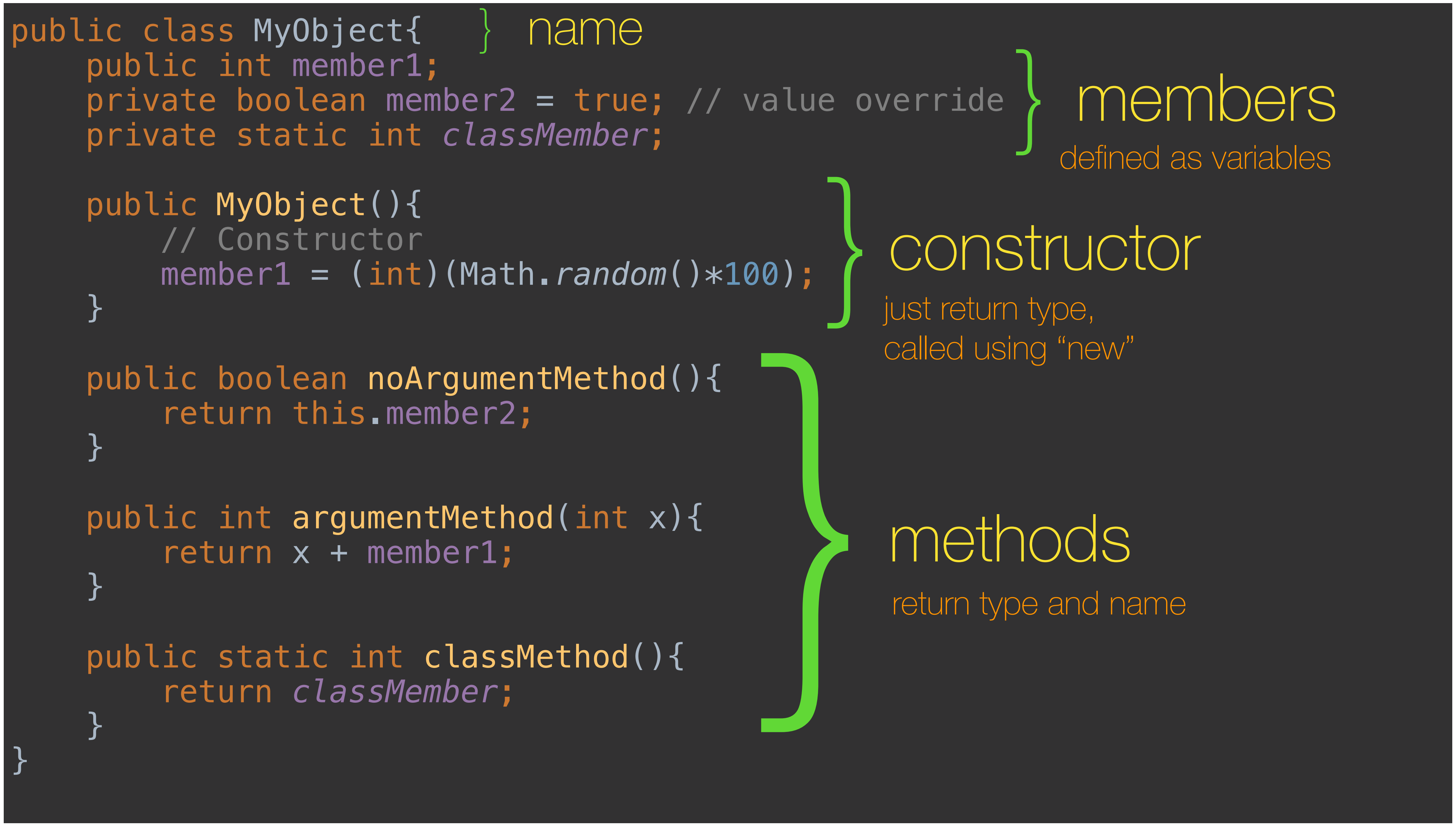


Class

Object

Class structure

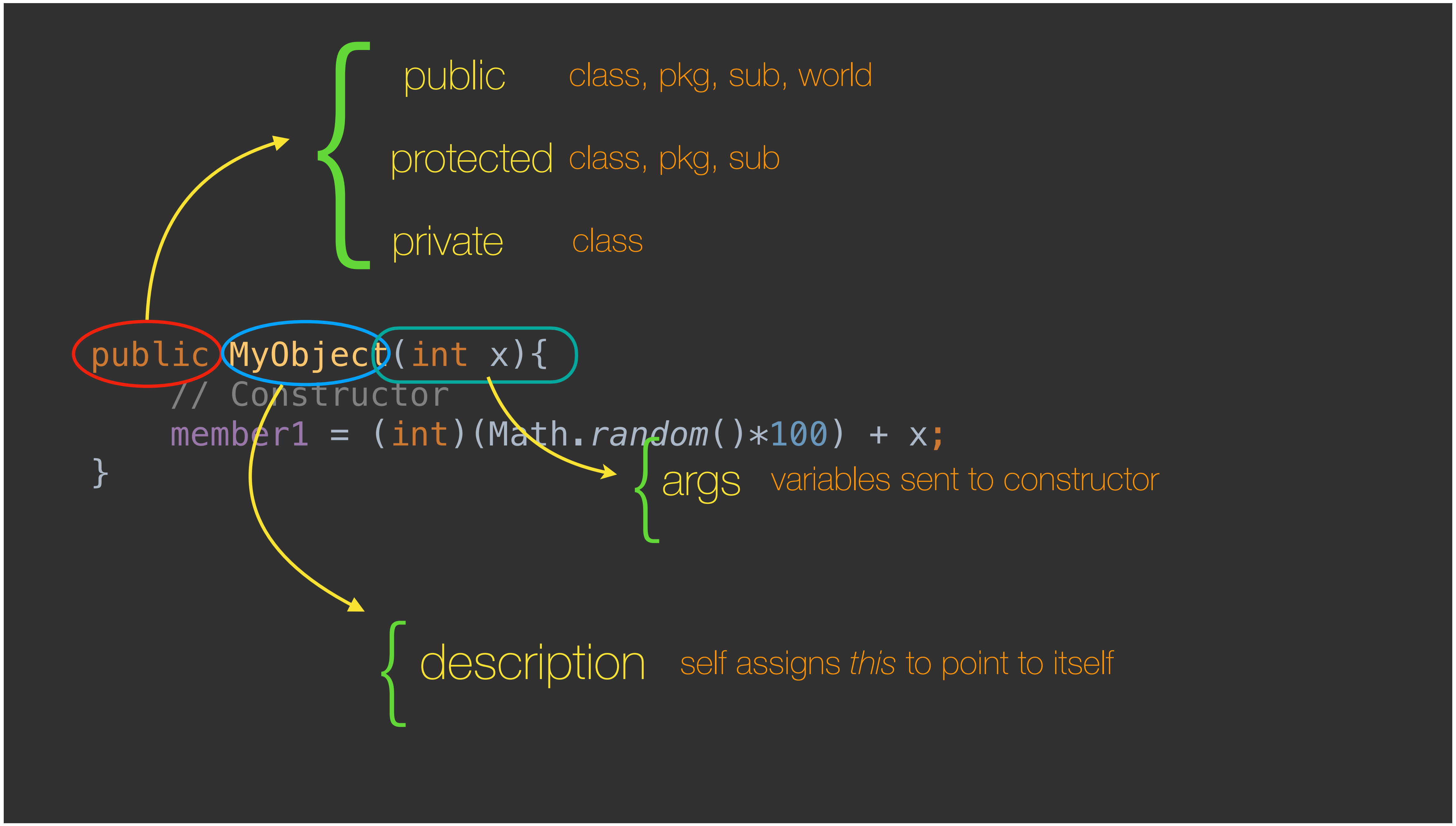
members, methods



```java
public class MyObject{
    public int member1;
    private boolean member2 = true; // value override
    private static int classMember;

    public MyObject(){
        // Constructor
        member1 = (int)(Math.random()*100);
    }

    public boolean noArgumentMethod(){
        return this.member2;
    }

    public int argumentMethod(int x){
        return x + member1;
    }

    public static int classMethod(){
        return classMember;
    }
}
```

Constructor

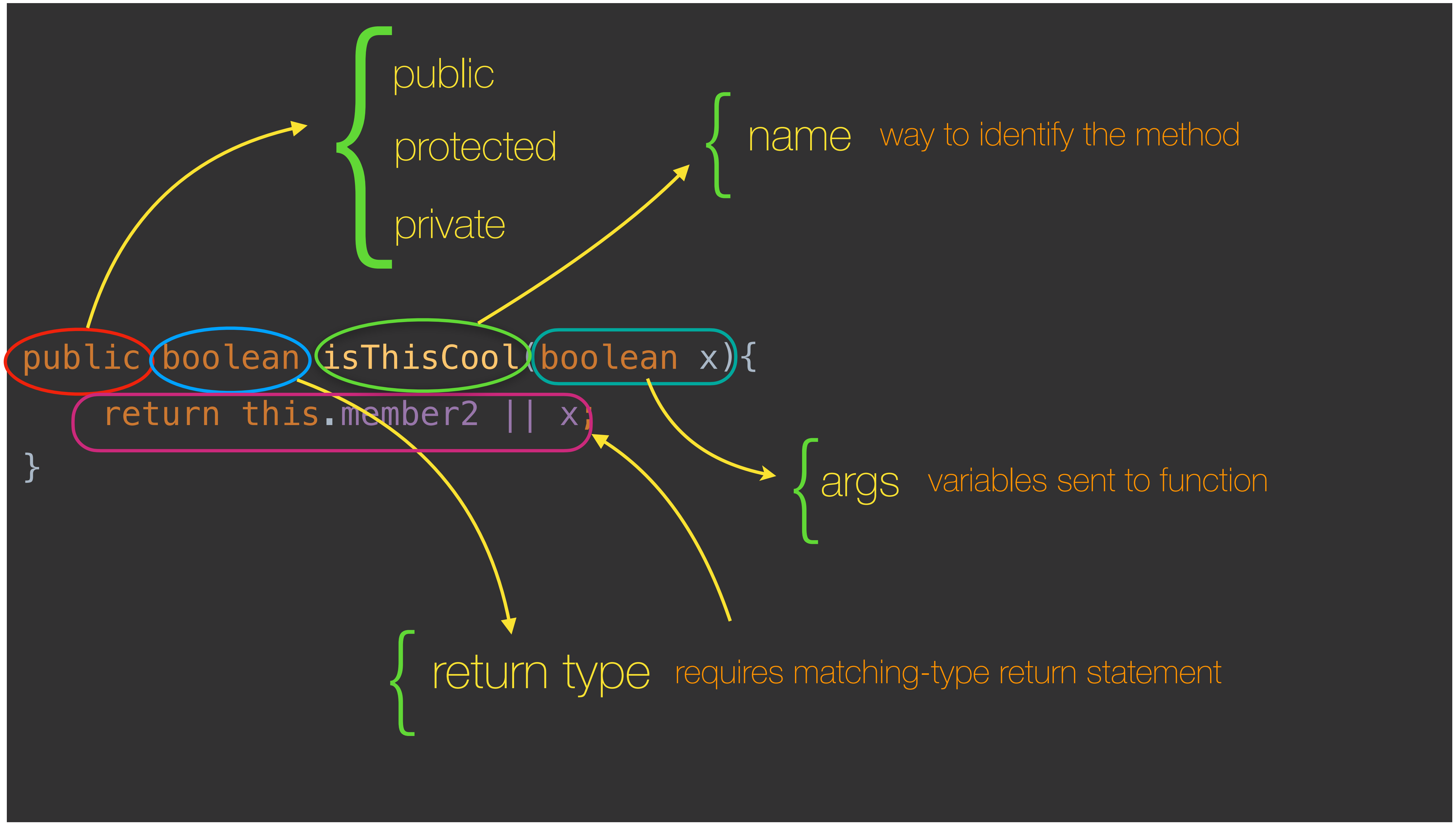
elements of a constructor


public class, pkg, sub, world
protected class, pkg, sub
private class
args variables sent to constructor
description self assigns *this* to point to itself

```
public MyObject(int x){
    // Constructor
    member1 = (int)(Math.random()*100) + x;
}
```

Methods

elements of a method

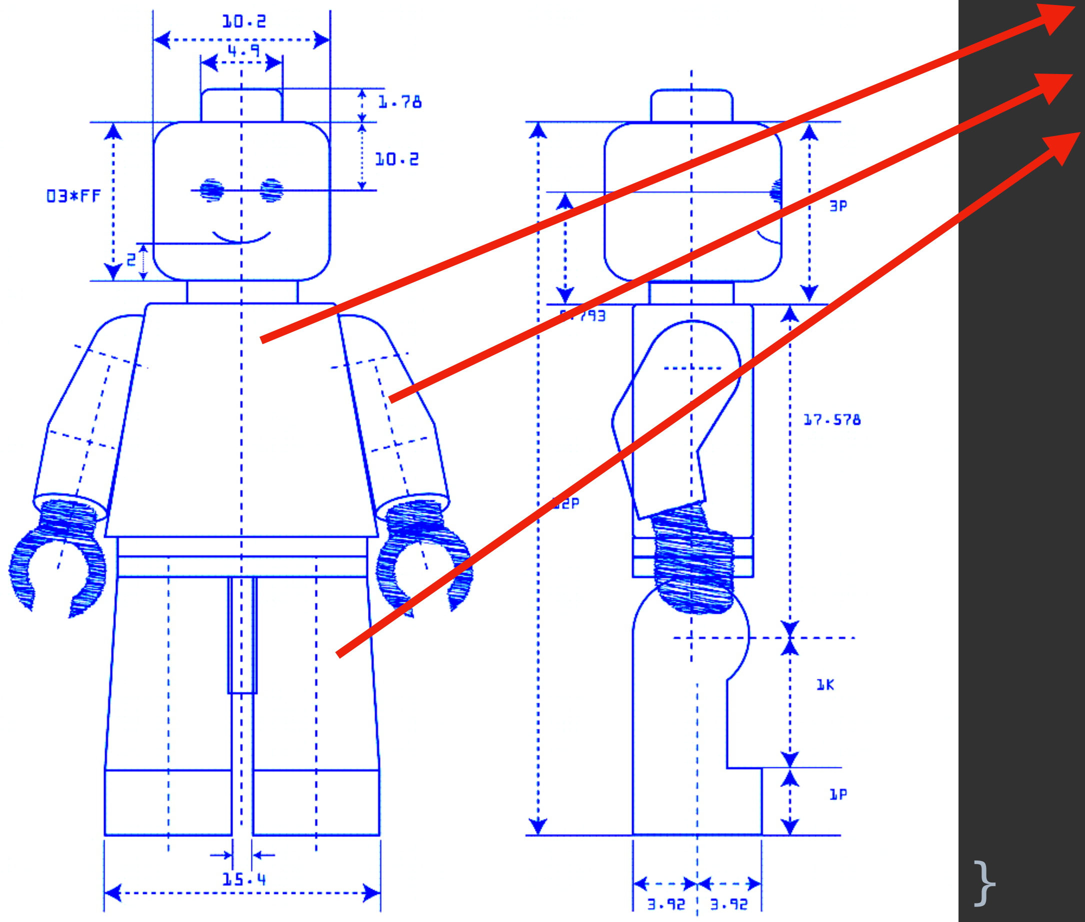


```
public boolean isThisCool(boolean x){
    return this.member2 || x;
}
```

public
protected
private
name way to identify the method
args variables sent to function
return type requires matching-type return statement

Classes

Show me all the blueprints

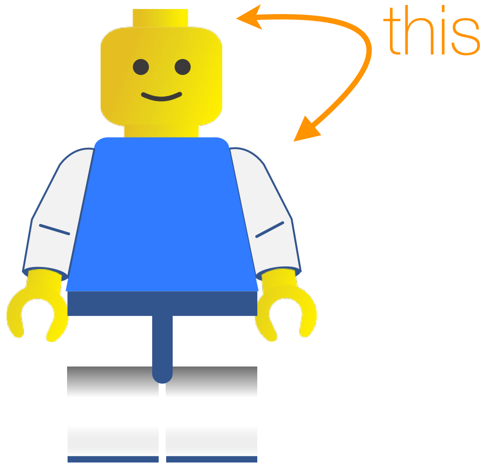


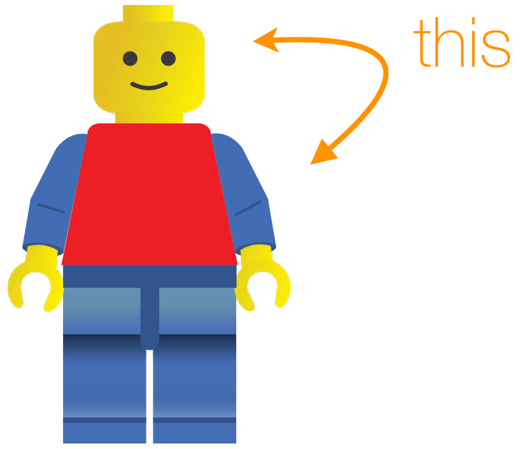
```
public class LegoMan{
    private String shirtColor = "blue";
    private String armColor = “white";
    private String pantsColor = “white”;

    public LegoMan(String shirtColor,
                   String pantsColor,
                   String armColor)
    {
        this.pantsColor = pantsColor;
        this.shirtColor = shirtColor;
        this.armColor = armColor;
    }

    public LegoMan(){
    }
}
```

Class to Object

Building Objects

```
LegoMan lm1 = new LegoMan();


LegoMan lm = new LegoMan("red", "blue", “red");
```





***Upon instantiation, variable this is assigned to itself**

Given the class

```java
public class Square {

    double x;
    double y;
    double width = 1;

    public Square(double x_pos, double y_pos)
    {
        this.x = x_pos; // Assign x_pos to x
        this.y = y_pos; // Assign y_pos to y
    }

    public void setWidth(double mywidth){
        this.width = mywidth;
    }

    public void scale(double factor){
        this.width = this.width*factor;
    }

    public void rotate(double angle){
        this.x = x*Math.cos(angle) - y*Math.sin(angle);
        this.y = x*Math.cos(angle) + y*Math.cos(angle);
    }
}
```

just trust this math

When this code happens

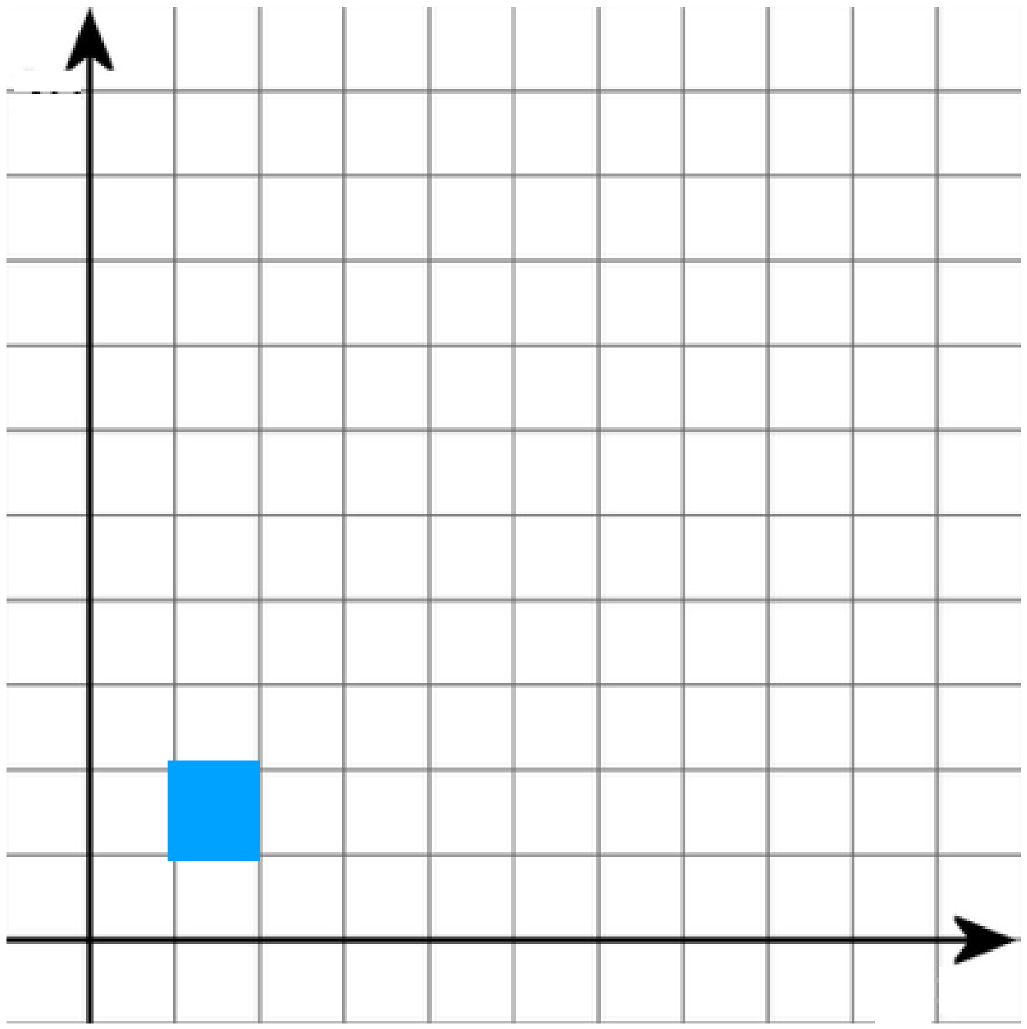
```
Square first = new Square(1,1);
```

Calls

```
public Square(double x_pos, double
y_pos)
{
    this.x = x_pos;
    this.y = y_pos;
}
```

Becomes

```
public Square(1.0, 1.0)
{
    this.x = 1.0;
    this.y = 1.0;
}
```



then if we did this

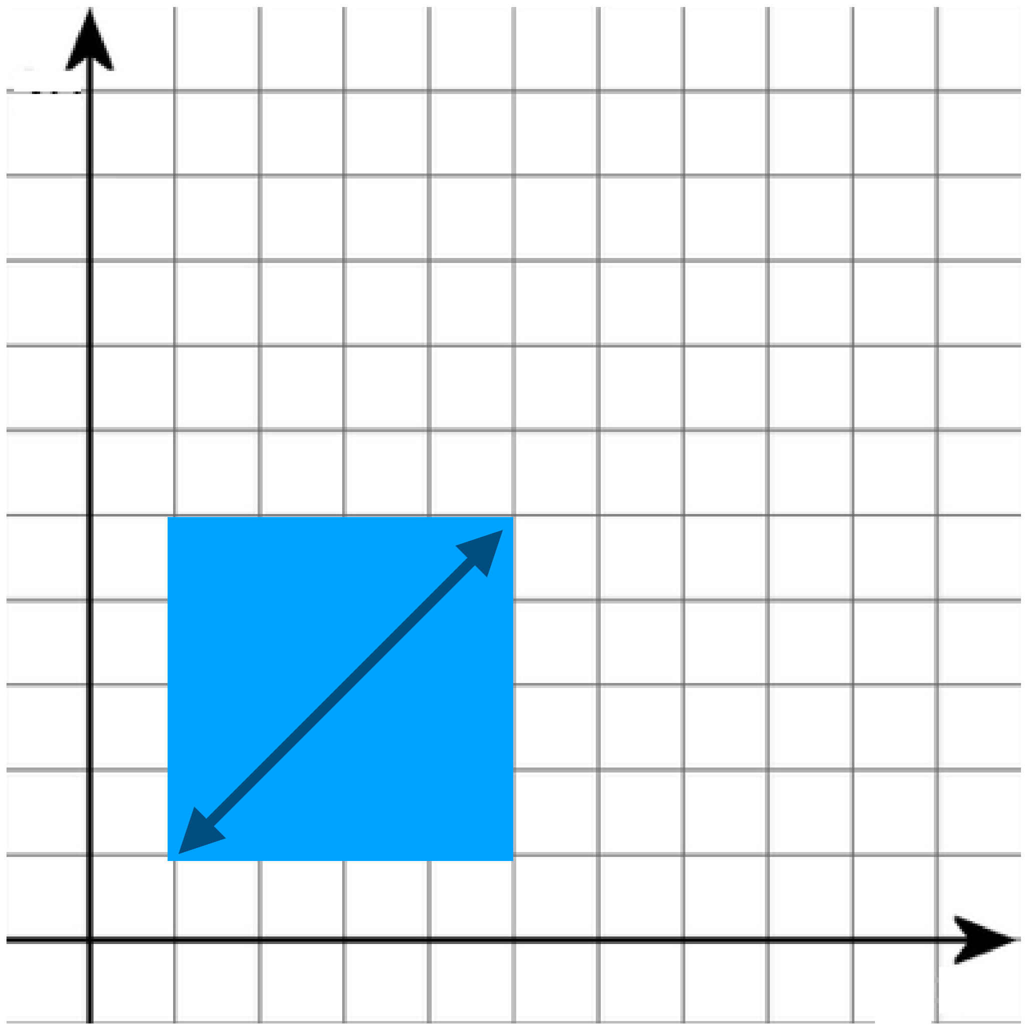
```
first.scale(4.0);
```

Calls

```
public void scale(double factor){
    this.width = this.width*factor;
}
```

Becomes

```
public void scale(4.0){
    this.width = 1.0 * 4.0;
}
```



what if we added

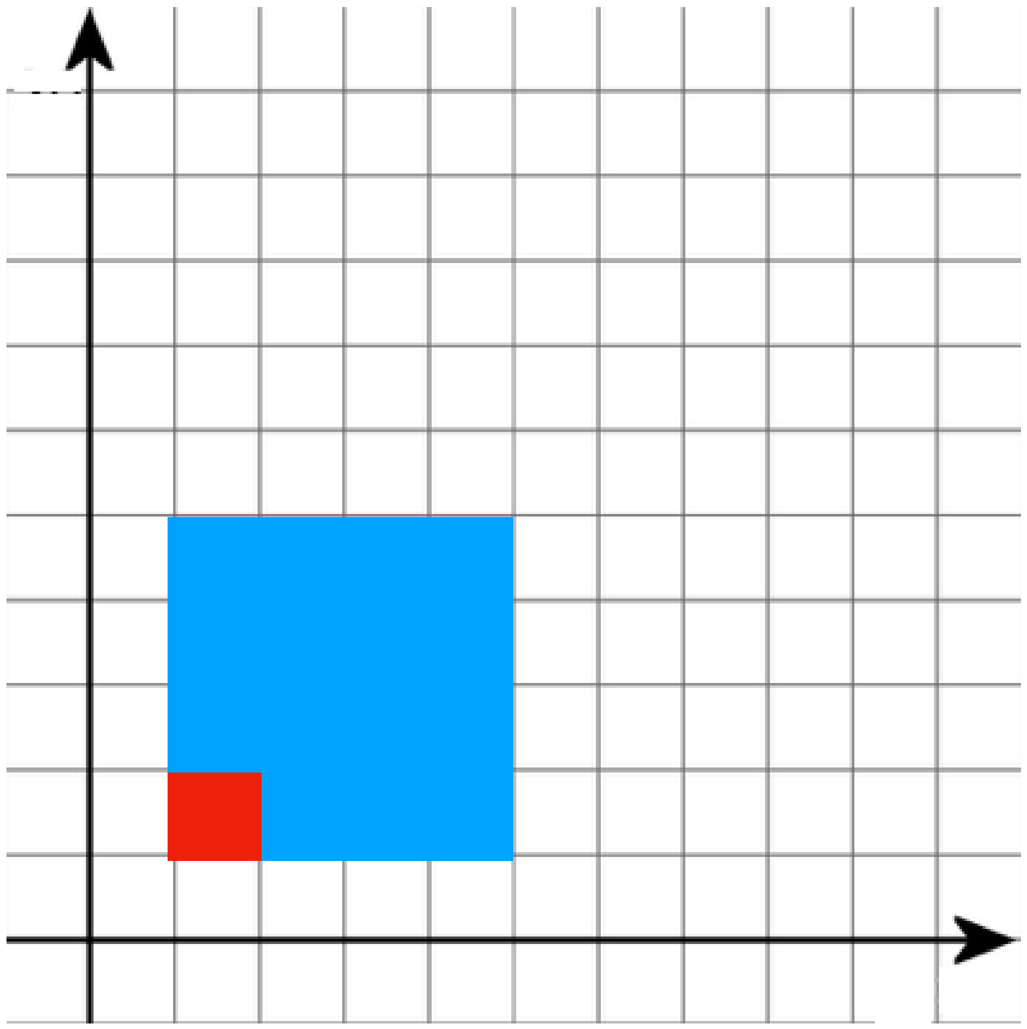
```
Square second = new Square(1, 1);
```

Calls

```
public Square(double x_pos, double
y_pos)
{
    this.x = x_pos; // Assign x_pos to x
    this.y = y_pos; // Assign y_pos to y
}
```

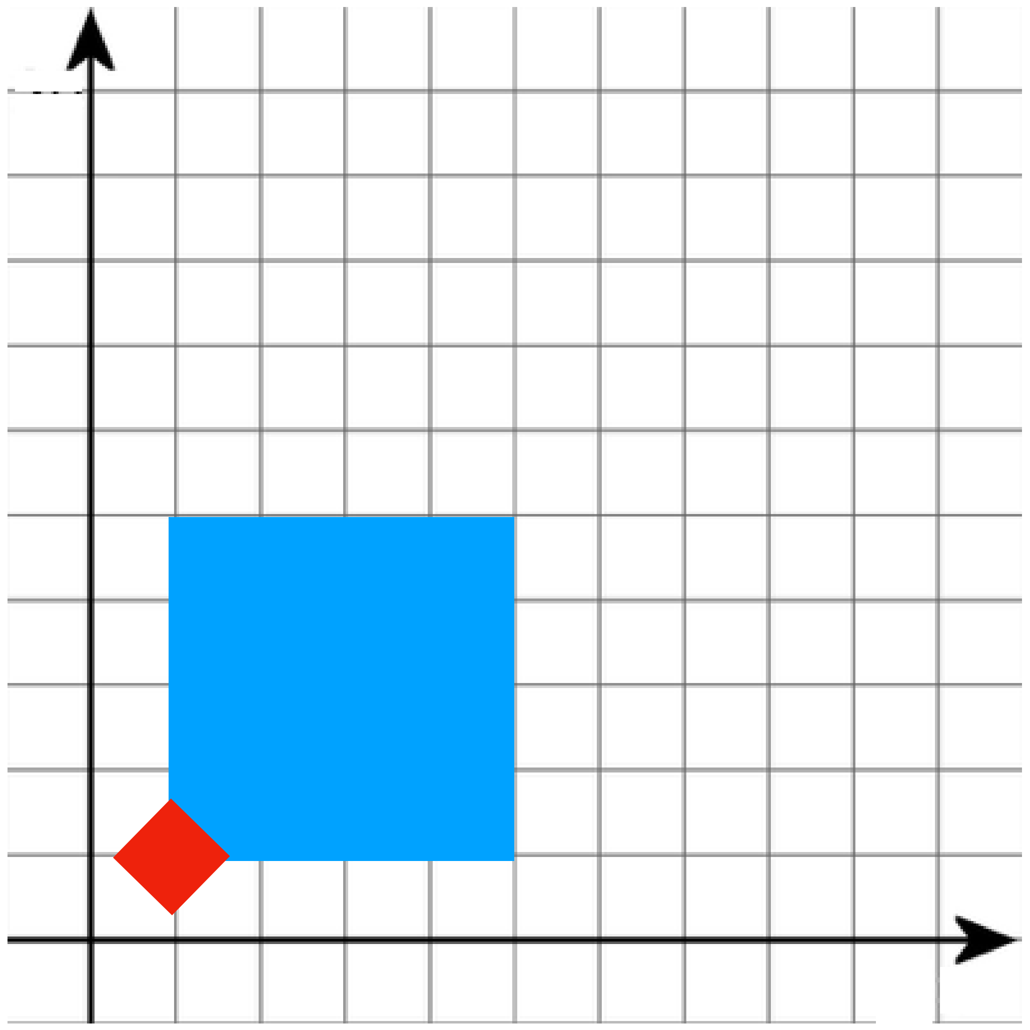
Becomes

```
public Square(1.0, 1.0)
{
    this.x = 1.0;
    this.y = 1.0;
}
```



What if?

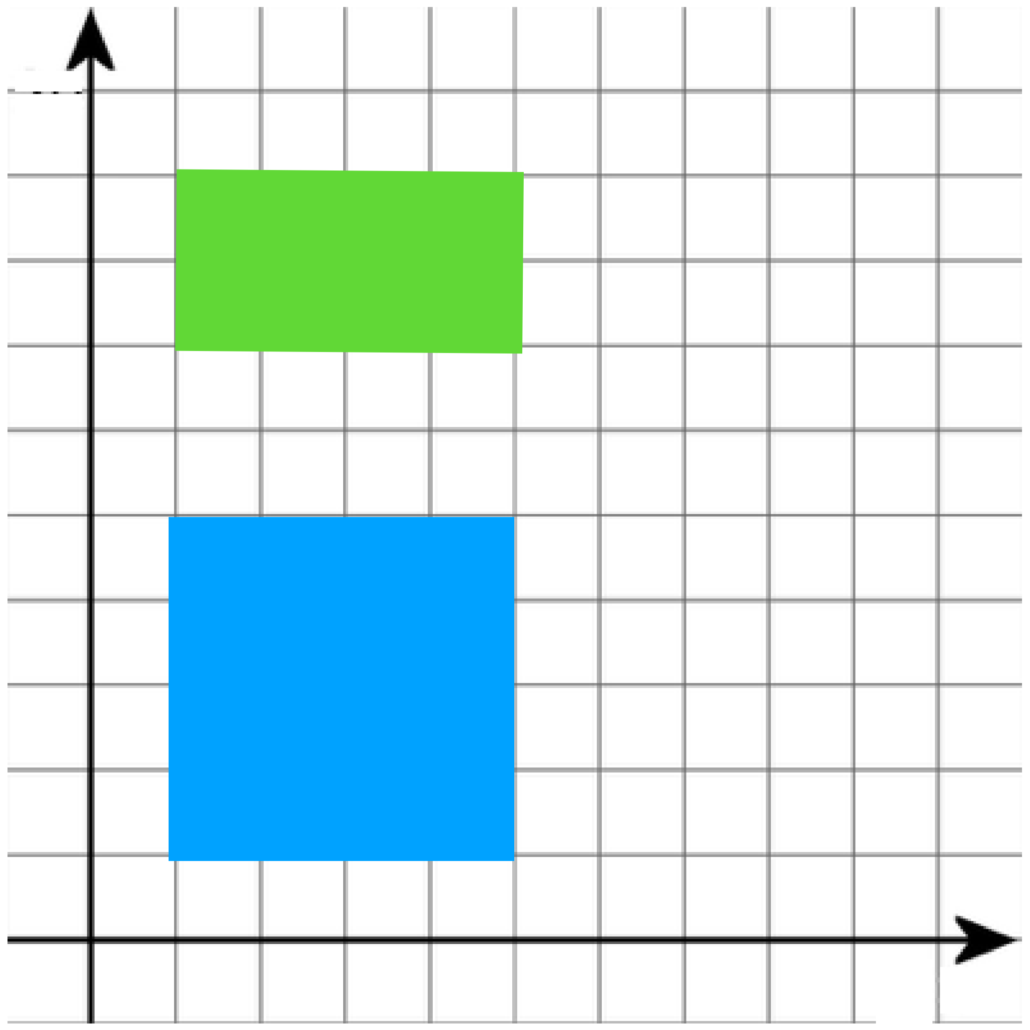
We wanted to rotate the second square by 45 degrees?



Rectangle

Isn’t a rectangle a square that has the same height and width?

What if we used the same base class to represent both?



Given the class

```java
public class Rectangle {

    double x;
    double y;
    double width = 1.0;
    double height = 1.0;

    public Rectangle(double x_pos, double y_pos) {
        this.x = x_pos; // Assign x_pos to x
        this.y = y_pos; // Assign y_pos to y
    }

    public void setWidth(double mywidth) {
        this.width = mywidth;
    }

    public void setHeight(double myheight) {
        this.height = myheight;
    }

    public void scale(double factor) {
        this.width = this.width * factor;
        this.height = this.height * factor;
    }

    public void rotate(double angle) {
        this.x = x * Math.cos(angle) - y * Math.sin(angle);
        this.y = x * Math.cos(angle) + y * Math.cos(angle);
    }
}
```

When this code happens

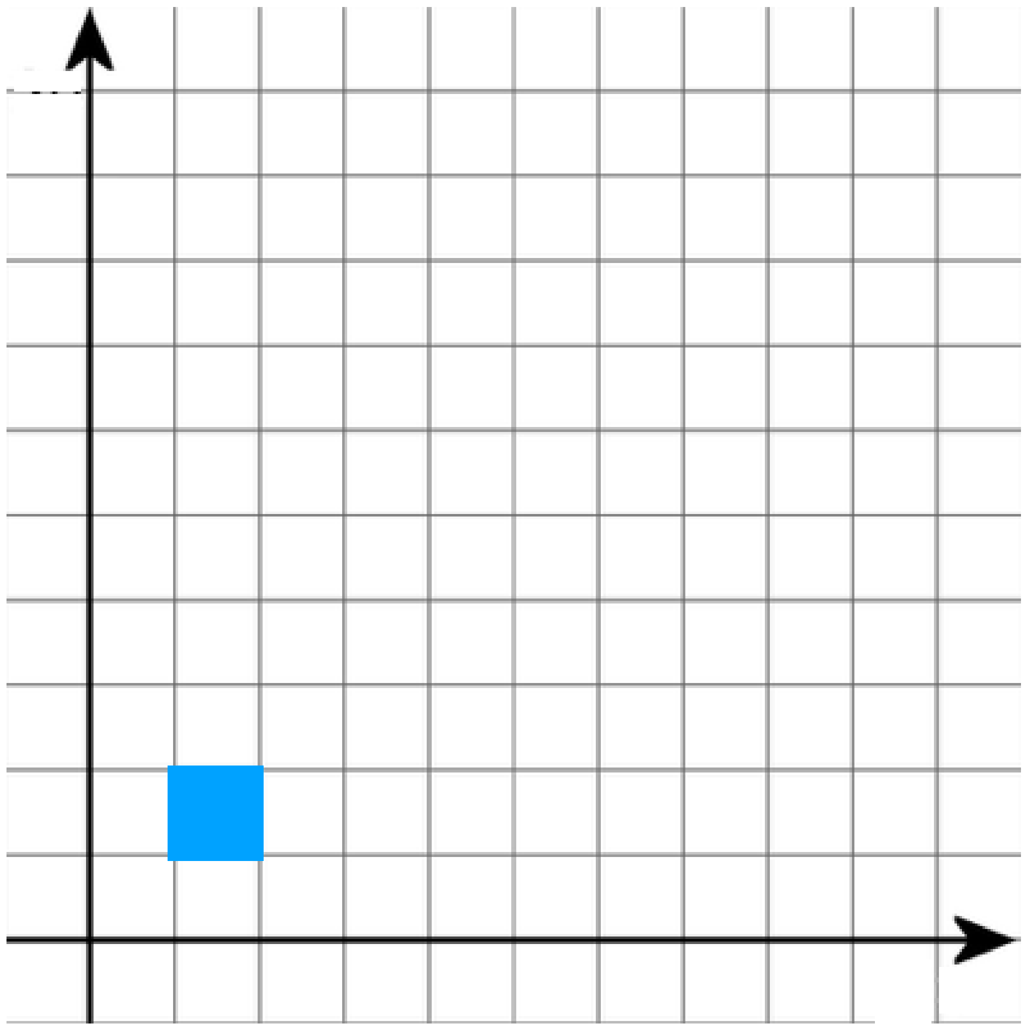
```
Rectangle first = new Rectangle(1,1);
```

Calls

```
public Rectangle(double x_pos, double
y_pos)
{
    this.x = x_pos;
    this.y = y_pos;
}
```

Becomes

```
public Rectangle(1.0, 1.0)
{
    this.x = 1.0;
    this.y = 1.0;
}
```



When this code happens

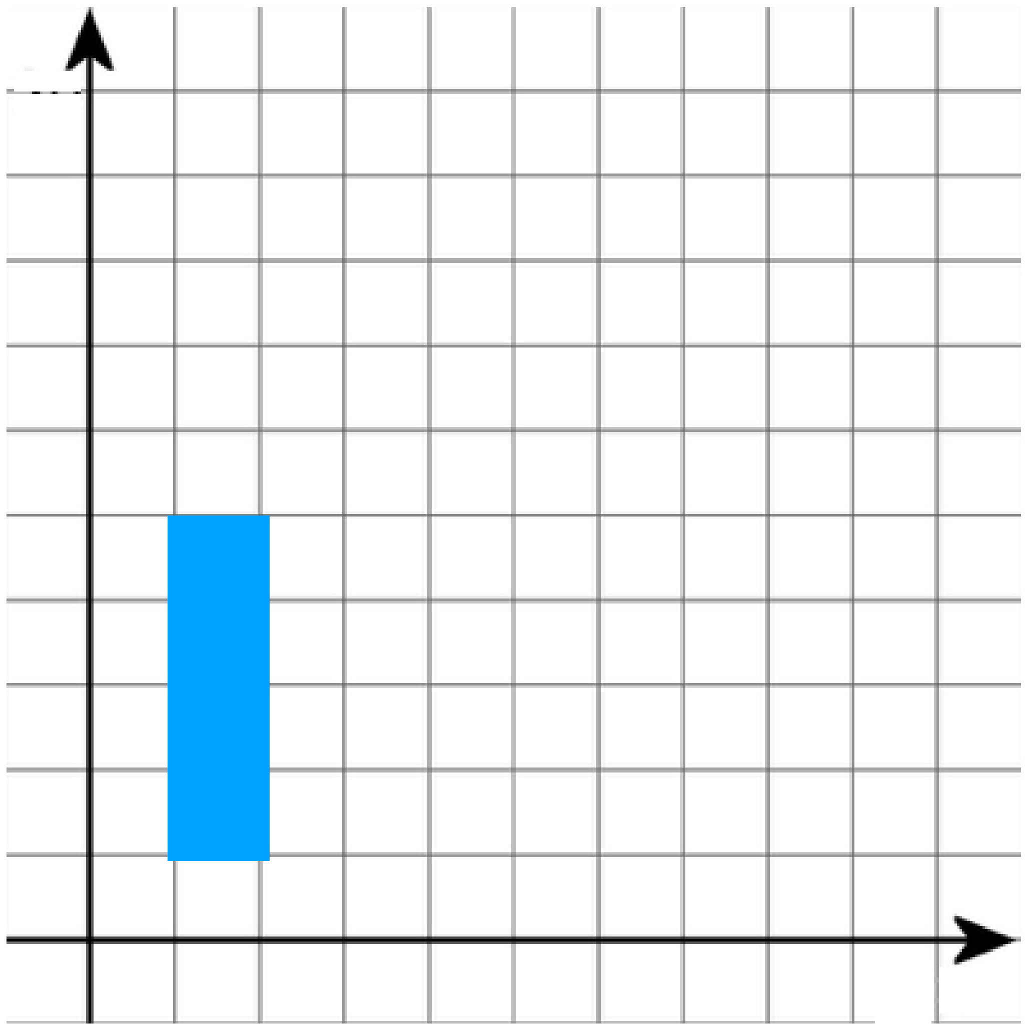
```java
first.setHeight(4.0);
```

Calls

```java
public void setHeight(double myheight)
{
    this.height = myheight;
}
```

Becomes

```java
public void setHeight(4.0) {
    this.height = 4.0;
}
```



So for Square

Notice that scale and rotate are not included

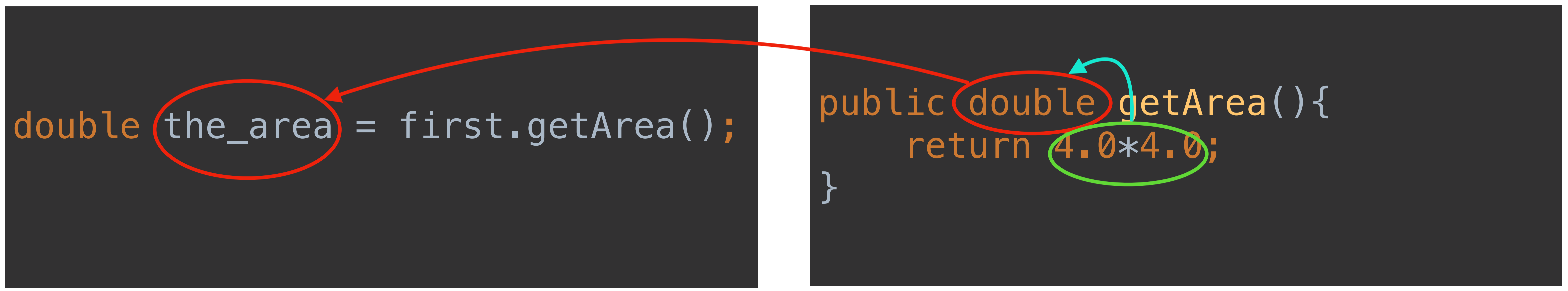
```
public class Square extends Rectangle {
    public Square(double x_pos, double y_pos){
        super(x_pos, y_pos);
    }

    public void setWidth(double mywidth) {
        this.width = mywidth;
        this.height = mywidth;
    }

    public void setHeight(double myheight) {
        this.height = myheight;
        this.width = myheight;
    }
}
```

Methods returning values

Values are sent back

```java
public double getArea(){
    return this.width*this.height;
}
```

```java
Square first = new Square(1,1);
first.setHeight(4);
double the_area = first.getArea();
```



```java
double the_area = first.getArea();
```

```java
public double getArea(){
    return 4.0*4.0;
}
```

All together now

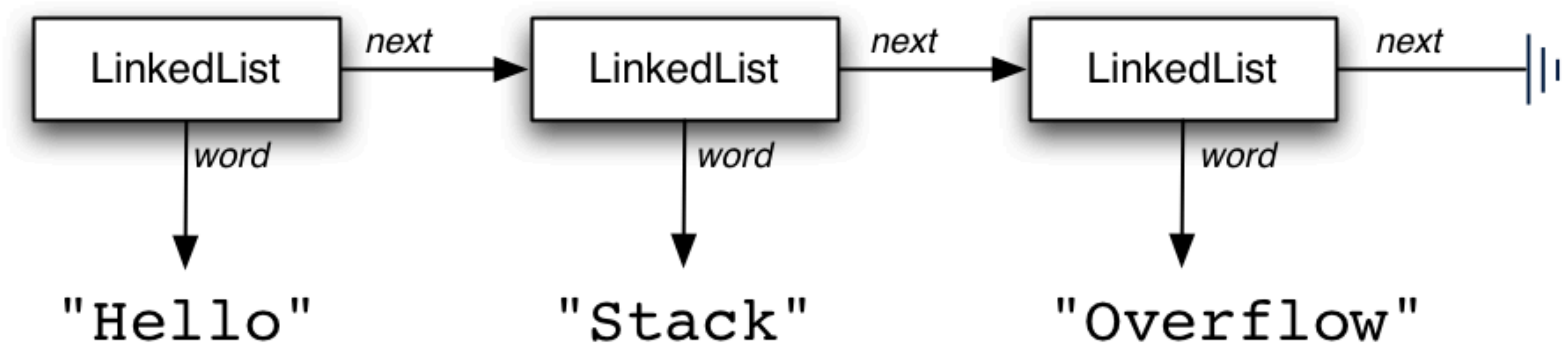
A Singleton

Just one and one only

```
public class Singleton{

}
```

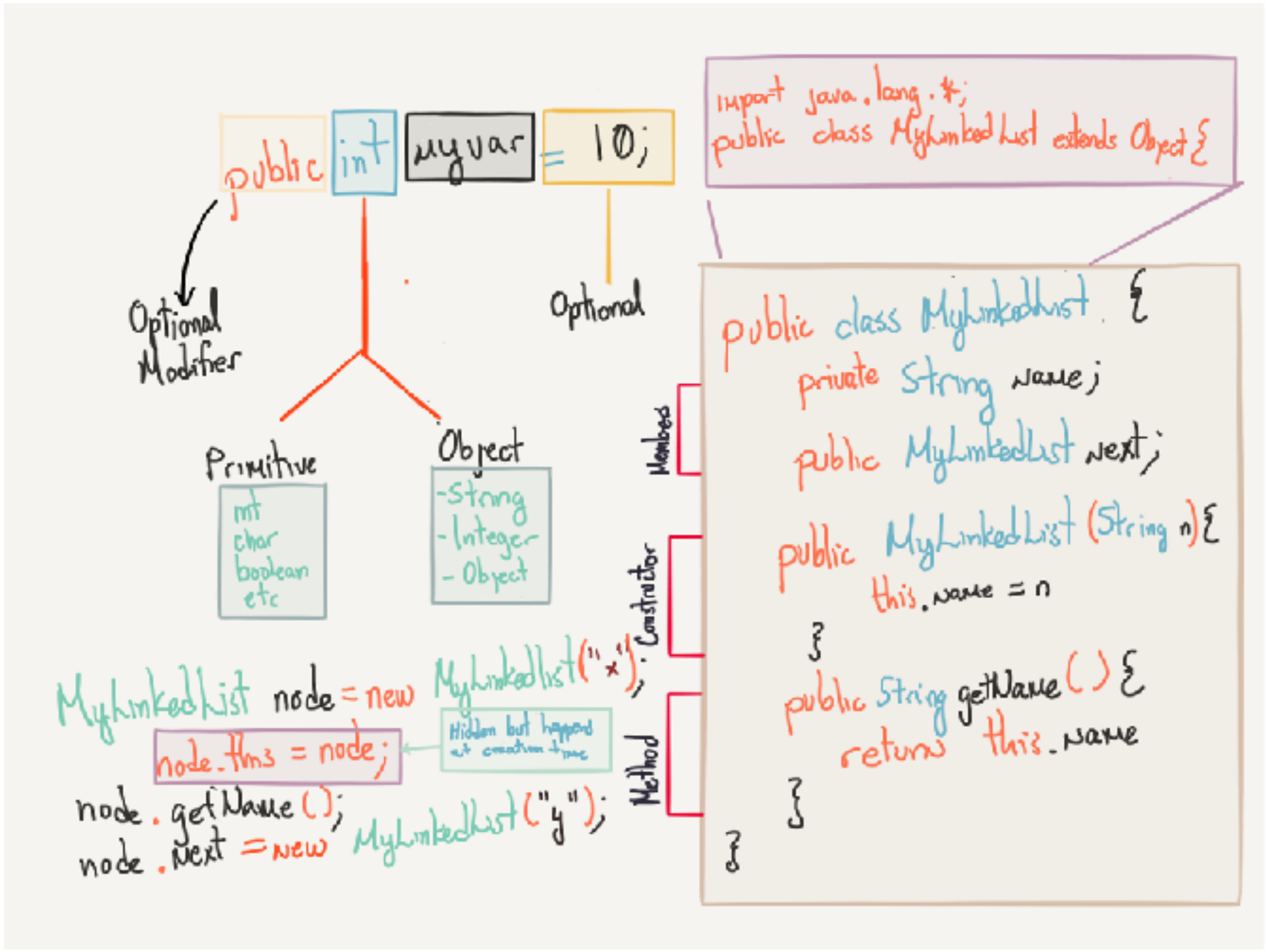
LinkedLists

Element knows its value and it knows about the next,



Before jumping to code

Lets understand the what a LinkedList class would look like



```
public int myvar = 10;
```

```
import java.lang.*;
public class MyLinkedList extends Object {
```

```
public class MyLinkedList {
    private String name;
    public MyLinkedList next;
    public MyLinkedList (String n){
        this.name = n
    }
    public String getName(){
        return this.name
    }
}
```

```
MyLinkedList node = new MyLinkedList("x");
node.this = node;
node.getName();
node.next = new MyLinkedList("y");
```

Homework

LinkedList

Implement a linked list in Java

```java
class LinkedList{

}
```