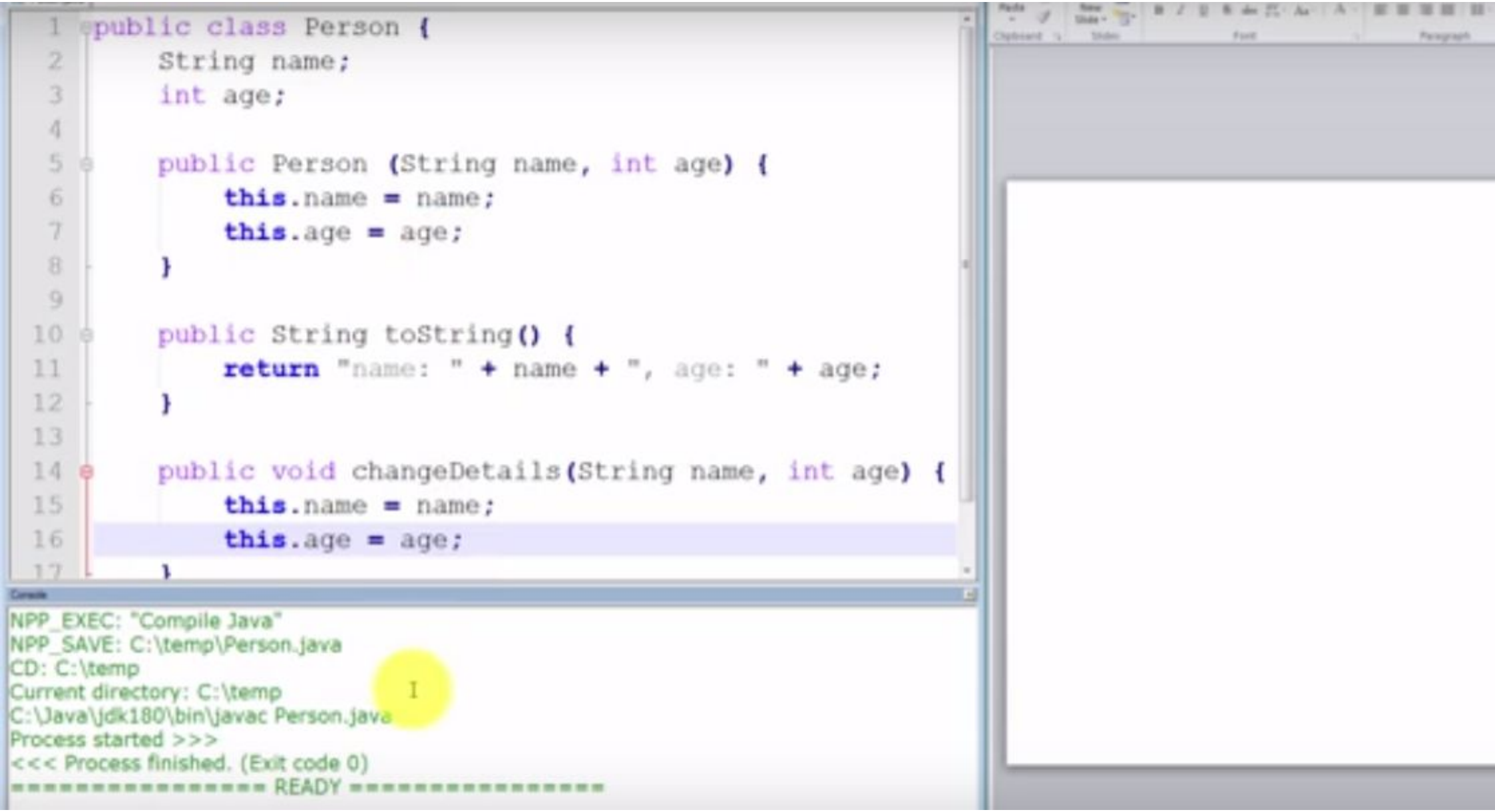
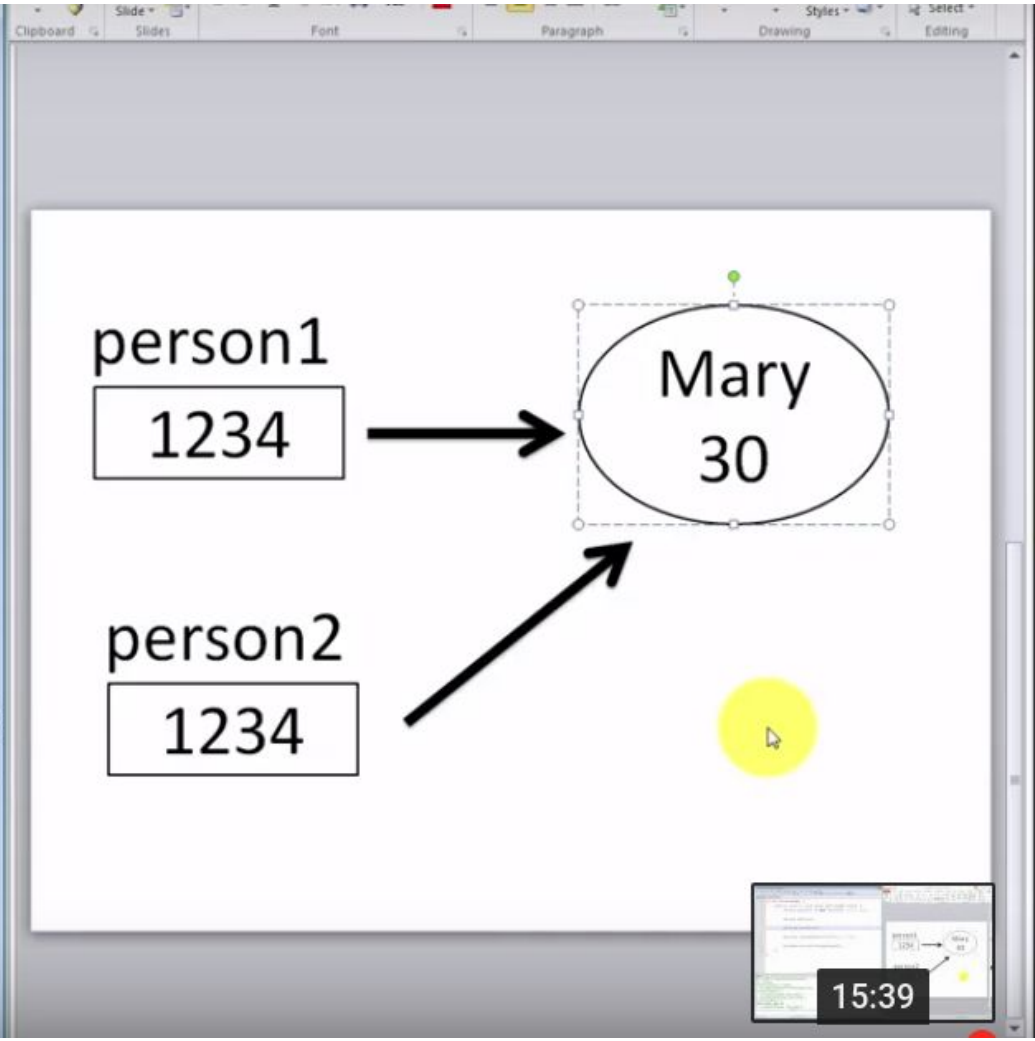

1 public class Person {
2 String name;
3 int age;
4
5 public Person (String name, int age) {
6 this.name = name;
7 this.age = age;
8 }
9
10 public String toString() {
11 return "name: " + name + ", age: " + age;
12 }
13
14 public void changeDetails(String name, int age) {
15 this.name = name;
16 this.age = age;
17 }
NPP_EXEC: "Compile Java"
NPP_SAVE: C:\temp\Person.java
CD: C:\temp
Current directory: C:\temp
C:\Java\jdk180\bin\javac Person.java
Process started >>>
<<< Process finished. (Exit code 0)
================ READY ================

```java
class PersonDemo {
    public static void main (String[] args) {
        Person person1 = new Person("John", 21);

        Person person2;

        person2 = person1;

        person2.changeDetails("Mary", 30);

        System.out.println(person1);
    }
}
```

```
Console
NPP_SAVE: C:\temp\PersonDemo.java
CD: C:\temp
Current directory: C:\temp
C:\Java\jdk180\bin\javac PersonDemo.java
Process started >>>
<<< Process finished. (Exit code 0)
C:\Java\jdk180\bin\java PersonDemo
Process started >>>
name: Mary, age: 30
<<< Process finished. (Exit code 0)
```



Defining a method with an object as a parameter

- When defining a method with a parameter specify the *data type* of each parameter.
- In this case, the *type* of the parameter is a Rectangle

```
public static void displayRectangle (Rectangle rect)
{
        System.out.println(rect.getLength());
        System.out.println(rect.getWidth());
}
```

- **rect** can call all of the methods defined in the Rectangle class.

Calling a method and passing an object

- When calling a method put the name of the object within the parenthesis

```
public static void main(String [] args)
{
    //Create a Rectangle object named r1
    Rectangle r1 = new Rectangle (5,7);

    //Call the displayRectangle method
    displayRectangle(r1);
}
```

Main Idea

- When an object is passed to a method, the memory address is being passed, not the data stored in it.

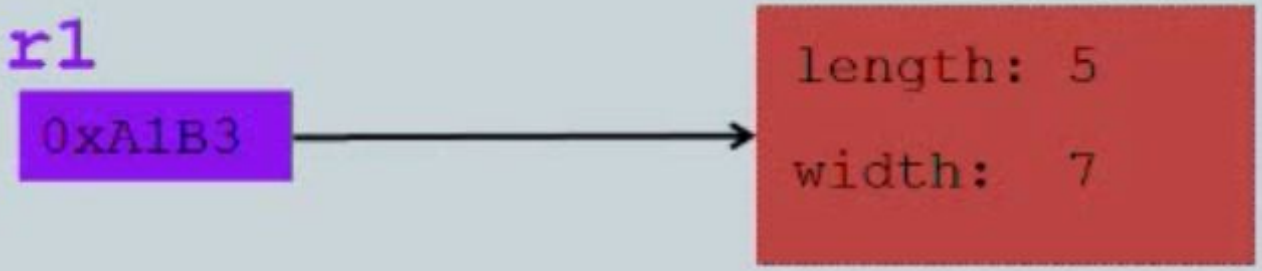
Passing Objects as Arguments to Methods - Blanton

Declaring an Object

- Create an object of a Rectangle class

```
Rectangle r1 = new Rectangle (5,7);
```

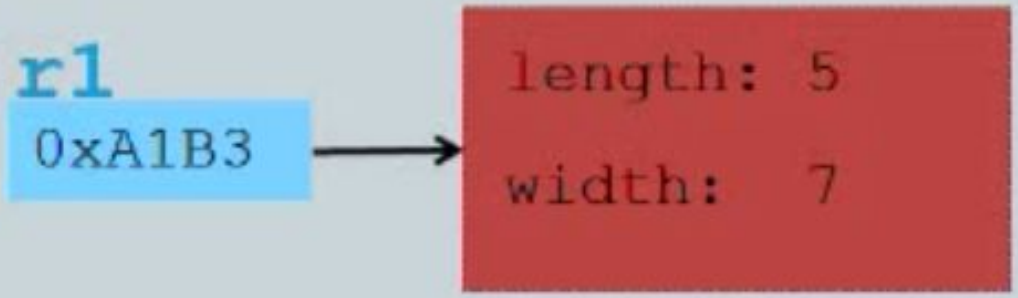
- **r1** holds the memory address of the object



Passing an object as a parameter

- The *address* stored in the argument r1 is sent to the parameter, rect

displayRectangle(r1);



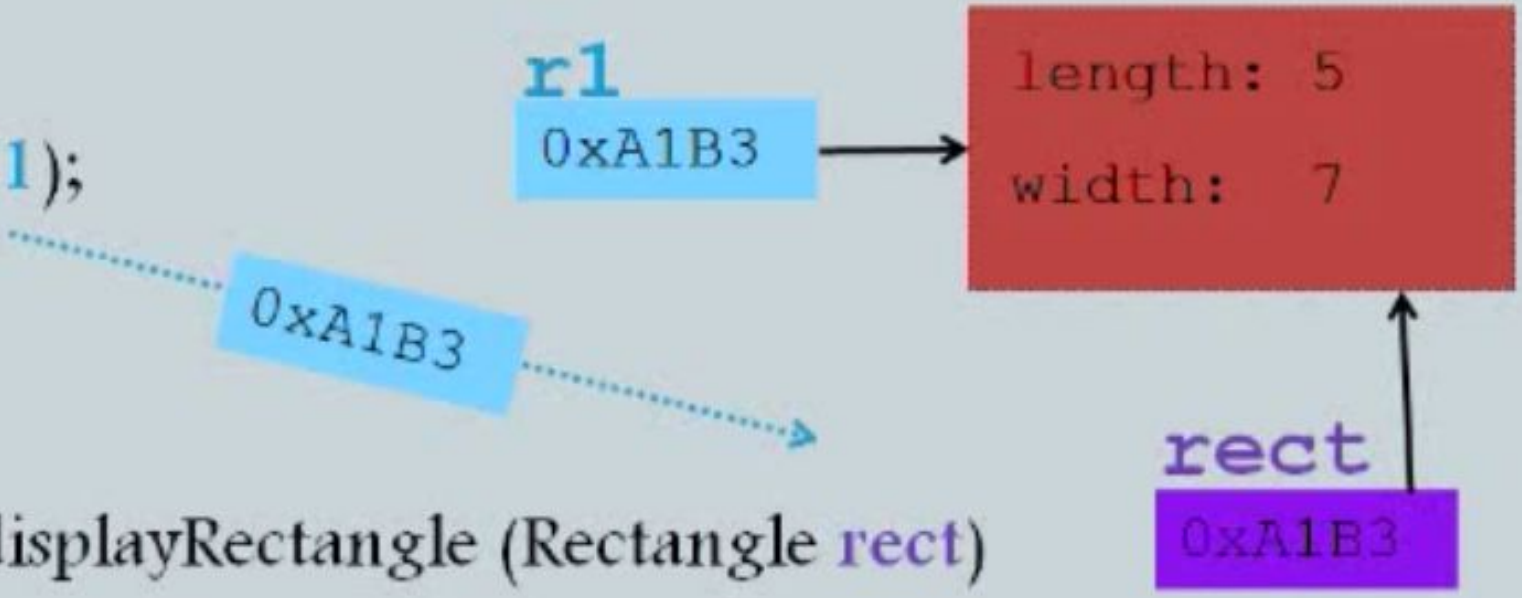

```
public static void displayRectangle (Rectangle rect)
```

- Both r1 and rect refer to the same Rectangle object

Passing an object as a parameter

- The *addresses*stored in the argument r1 is sent to the parameter, rect

displayRectangle(r1);



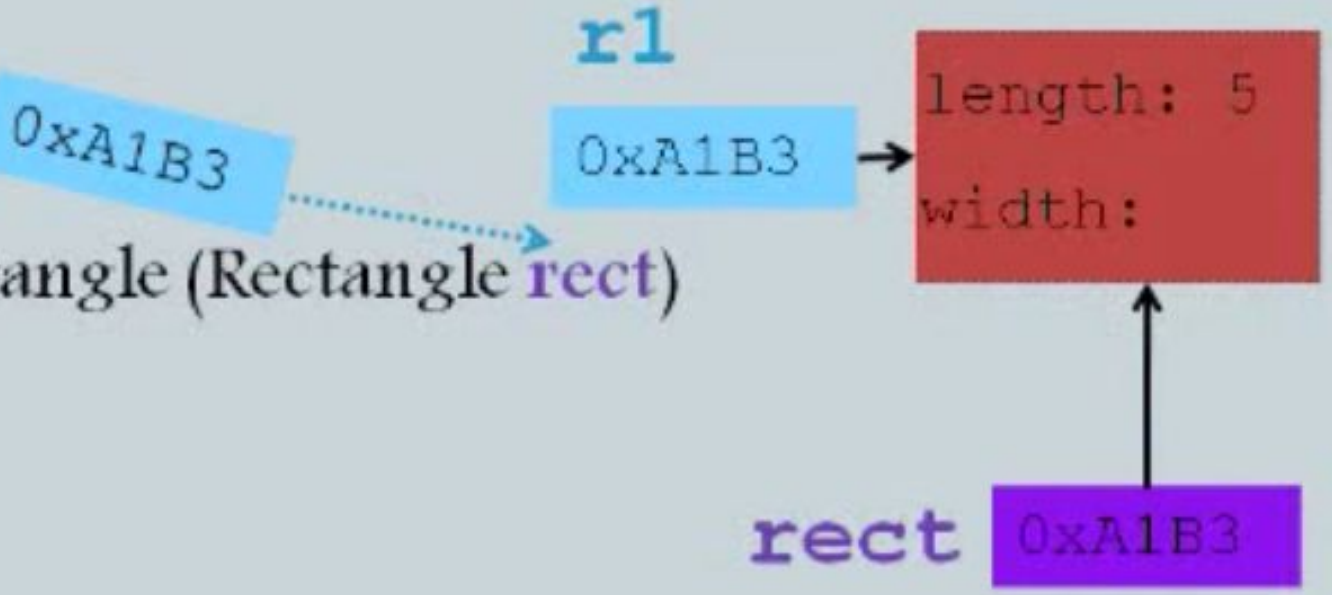
public static void displayRectangle (Rectangle rect)

- Both r1 and rect refer to the same Rectangle object

Passing an object as a parameter

- When a method receives an object reference as an argument, it may modify the contents of the object

```
setRectangle(r1);

public static void setRectangle (Rectangle rect)
{
    rect.setLength(6);
    rect.setWidth(3);
}
```

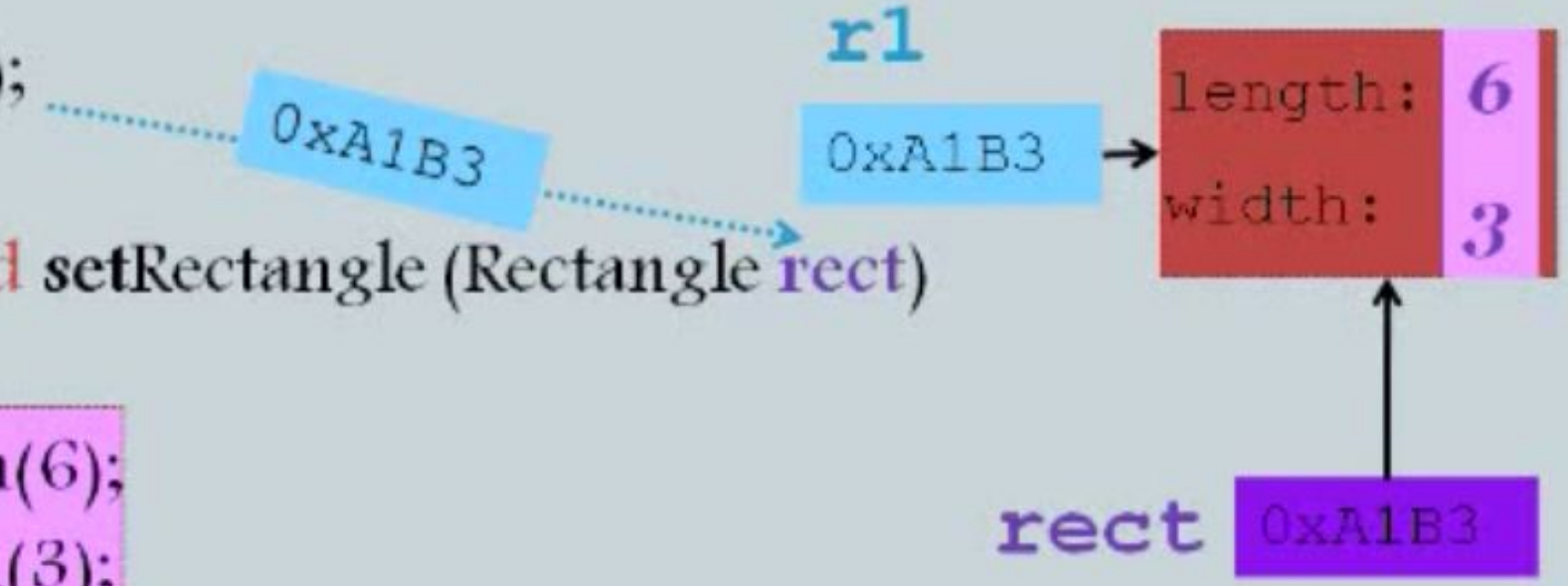


- Changes made to the object referenced by rect will also update r1

Passing an object as a parameter

- When a method receives an object reference as an argument, it may modify the contents of the object

```
setRectangle(r1);

public static void setRectangle (Rectangle rect)
{
    rect.setLength(6);
    rect.setWidth(3);
}
```

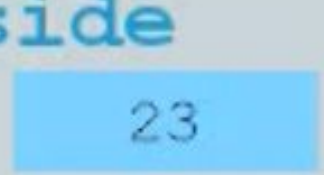


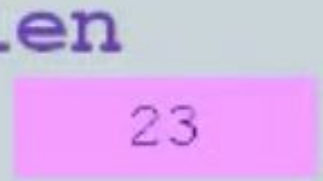
- Changes made to the object referenced by rect will also update r1

Passing Primitive Data Types

- Changing a parameter of a primitive data type will *not* change the argument.

```
public static void main (String [] args)
{
   double side= 23;
   showSquare(side) ;
}

public static void showSquare   (double len)
{
   len = len*len;
   System.out.println(len + "sq ft");
}
```

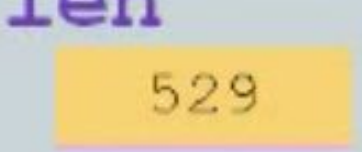




Passing Primitive Data Types

- Changing a parameter of a primitive data type will *not* change the argument.

```
public static void main (String [] args)
{
   double side= 23;
   showSquare(side) ;
}

public static void showSquare   (double len)
{
   len = len*len;
   System.out.println(len + "sq ft");
}
```



Key Points

- It is useful to pass an object as an argument to a method because when doing so you pass the memory address to the method instead a copy of a value. This means that you can update the object in another location and it will apply to the method.
- When writing a method that receives an object as an argument, do not accidentally write code that modifies the object.