Object Oriented Programming: Global (Instance) Variables

Mr. Neat
Java

Graphics Library: Horstmann.com

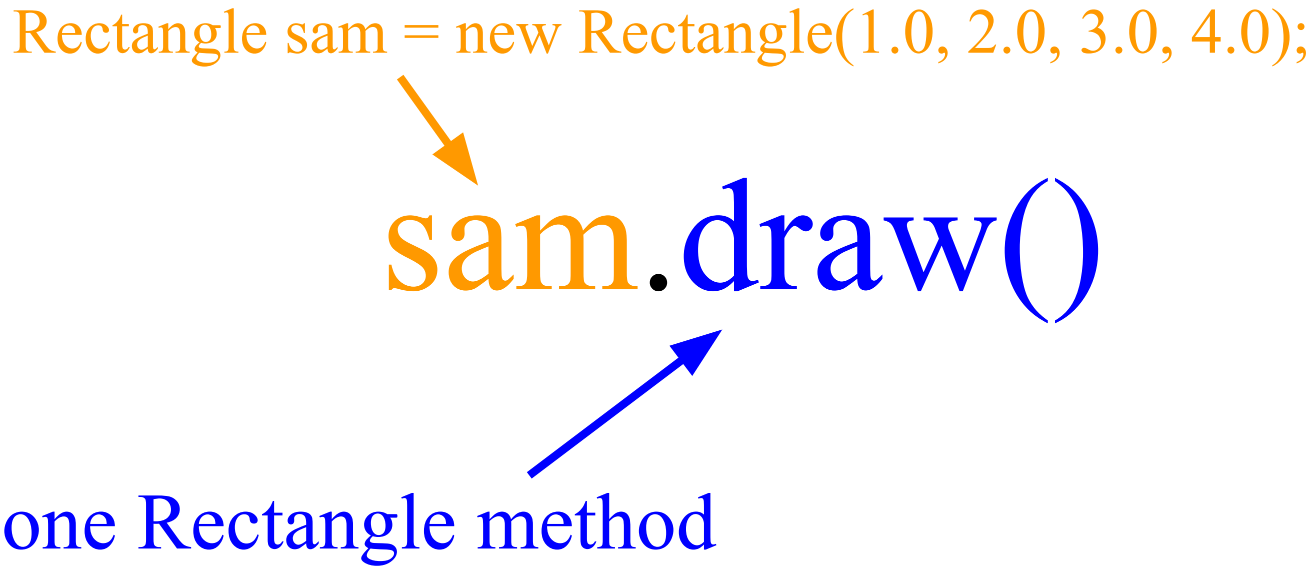
Application
Programming
Interface

API

http://horstmann.com/sjsu/graphics/api/index.html

Graphics Library: Calling Methods - Review

made from a Rectangle constructor:



```
Rectangle sam = new Rectangle(1.0, 2.0, 3.0, 4.0);
```

```
sam.draw()
```

one Rectangle method

Calling Java Methods

What if we wanted our Rectangle sam to move?

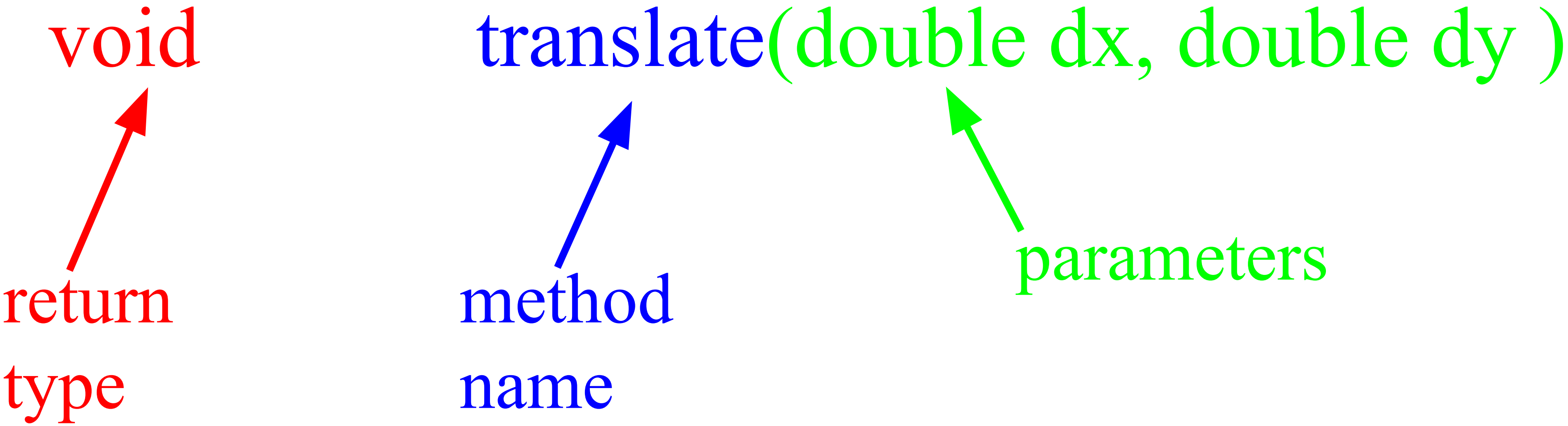
What method from the Rectangle class would do this?

Calling Java Methods

Do it!

(use base code in this folder)

Calling Java Methods



void translate(double dx, double dy)

return type

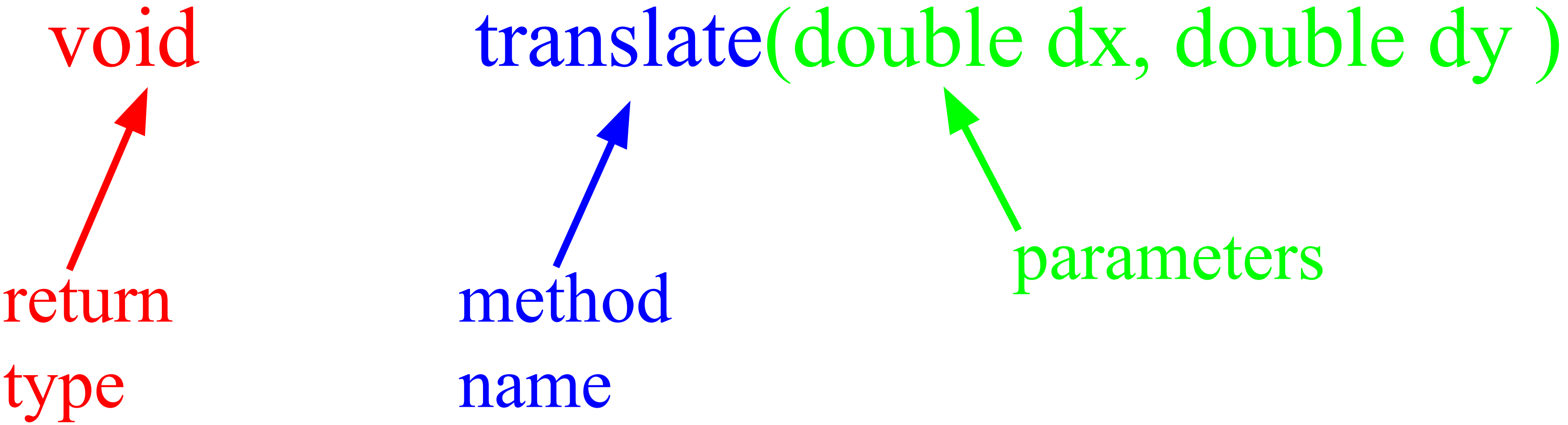
method name

parameters

btw, translate is a mutator method….
(changes the object)

Calling Java Methods

dx and dy tell the Rectangle how much to translate
in the x and y directions



void translate(double dx, double dy)

return type

method name

parameters

btw, translate is a mutator method….
(changes the object)

Calling Java Methods

```
public static void main(String args[])
{
        // following line is necessary for onMouseClick, don't change

        MouseController mC = new MouseController(Canvas.getInstance(),new starter());

        // put your code here:

        Rectangle m = new Rectangle(50,100,300,300);
        m.draw();
        m.translate(5.0,0.0);
}
```

What happened?

Calling Java Methods
Java is GREAT @ event driven actions!

```
public static void main(String args[])
{


}
public void onMouseClick(double x, double y)
{


}
```

whatever code is in between these { } executes one time, each time the mouse is clicked

Calling Java Methods

Do it!

(use base code in this folder)

Calling Java Methods

```
public static void main(String args[])
{
        // following line is necessary for onMouseClick, don't change

        MouseController mC = new MouseController(Canvas.getInstance(),new starter());

        // put your code here:

        Rectangle m = new Rectangle(50,100,300,300);
        m.draw();
}

public void onMouseClick(double x, double y)
{

        m.translate(5.0,0.0);


}
```

What happened?

Calling Java Methods

This is a RUNTIME error!

Can you tell what type *m* is from this code segment?

```
public void onMouseClick(double x, double y)
{

    m.translate(5.0,0.0);

}
```

Calling Java Methods

Introducing...Global Variables (Instance Variables)

Global variables are variables (for example, *m*) that are defined outside of any method, but inside of a class definition.

Introducing...Global Variables (Instance Variables)

Every variable has scope. Its scope is dependent on where it is defined.

```
public static void main(String args[])
{
    // following line is necessary for onMouseClick, don't change

    MouseController mC = new MouseController(Canvas.getInstance(),new starter());

    // put your code here:

    Rectangle m = new Rectangle(50,100,300,300);
    m.draw();
}
```

scope for *m* is main() in this program

Introducing...Global Variables (Instance Variables)

```
public class starter implements InputControl
{
        static Rectangle m;

        public static void main(String args[])
        {
                MouseController mC = new MouseController(Canvas.getInstance(),new starter());
                m = new Rectangle(50,100,300,300);
                m.draw();
        }
        public void onMouseClick(double x, double y)
        {
                m.translate(5.0,0.0);
        }
}
```

- move Rectangle *m* outside of the method, but in the class definition
- now the scope of *m* is the whole class
- both *main* and *onMouseClick* can use *m*
- note *static* is required since *main* is *static* (more later)

Introducing...Global Variables (Instance Variables)

When defining a new variable, how do you decide what its scope is?

If only the method needs to know about it, define it within the method. But if another method needs to use it too or know about it, define it as a global variable.

Next Lab...

Make your Rectangle object move each time the mouse is clicked.