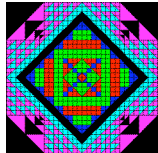


Inheritance

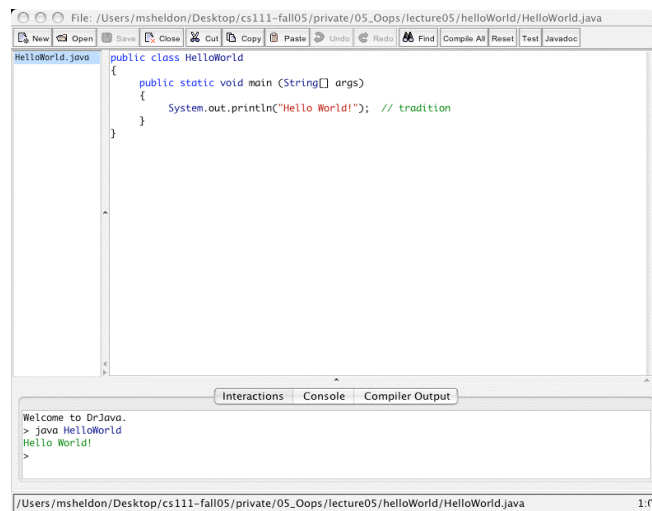
Methods of modularity



CS111 Computer Programming

Department of Computer Science
Wellesley College

Our first application



```
File: /Users/msheldon/Desktop/cs111-fall05/private/05_Oops/lecture05/helloWorld/HelloWorld.java
New Open Save Close Cut Copy Paste Undo Redo Find Compile All Reset Test Javadoc
HelloWorld.java public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello World!"); // tradition
    }
}

Interactions Console Compiler Output
Welcome to DrJava.
> java HelloWorld
Hello World!
>

/Users/msheldon/Desktop/cs111-fall05/private/05_Oops/lecture05/helloWorld/HelloWorld.java 1:6
```

Inheritance 5-2

A closer look

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello World!"); // tradition
    }
}
```

main method declaration
executed when you type
> java HelloWorld

body of
main method →

12 character string
" H e l l o W o r l d ! "
(everything inside of quotes)

Inheritance 5-3

A variation

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.print("Hello "); // also tradition
        System.out.println("World!");
    }
}
```

Inheritance 5-4

String is a class too!

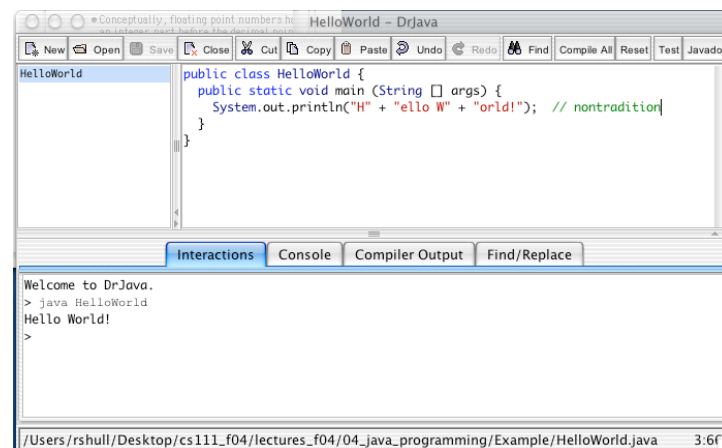
```
import java.awt.*;

public class HelloWorld2
{
    public static void main (String[] args)
    {
        String greeting = "Hello World!";
        System.out.println(greeting); // yet more tradition
        System.out.println(greeting.length()); // msg size
        System.out.println(greeting.toUpperCase()); // loud
        System.out.println(new String()); // ignore
    }
}
```

*Check out [String](#) contract on the cs111 web site.

Inheritance 5-5

String concatenation*



*Operators like + that have different meanings in different contexts are said to be **overloaded**.

Inheritance 5-6

Point class declaration



Point.java



Point.class

```
// Represents an (x,y) point
class Point
{
    public int x;
    public int y;

    public Point(int x_coord, int y_coord)
    {
        this.x = x_coord;
        this.y = y_coord;
    }

    public Point()
    {
        x = 0;
        y = 0;
    }

    public void greetings() // We made this one up
    {
        System.out.println("Hi");
    }
}
```

Inheritance 5-7

Testing our Point class



TestPoint.java



TestPoint.class

```
import java.awt.Color;

// Class used to test Point objects
public class TestPoint
{
    public static void main (String [] args)
    {
        Point p1 = new Point(5, 11);
        Point p2;
        p2 = new Point();

        p1.greetings();
        System.out.println(p1.x + "," + p1.y);

        int sum = p2.x + p2.y;
        System.out.println(sum);
    }
}
```

Inheritance 5-8

VerbosePoints



VerbosePoint.java



VerbosePoint.class

```
// Represents an (x,y) point with a message
class VerbosePoint extends Point
{
    public int x;
    public int y;

    public VerbosePoint(int x_coord, int y_coord)
    {
        this.x = x_coord;
        this.y = y_coord;
    }

    public VerbosePoint()
    {
        x = 0;
        y = 0;
    }

    public void greetings()
    {
        System.out.println(this + " says hello");
    }
}
```

Inheritance 5-9

But when we test our VerbosePoints

```
import java.awt.Color;

// Class used to test VerbosePoint objects
public class TestVerbosePoint
{
    public static void main (String [] args)
    {
        VerbosePoint p1 = new VerbosePoint(5, 11);
        VerbosePoint p2 = new VerbosePoint();

        p1.greetings();
        p2.greetings();
    }
}
/*****
// From VerbosePoint as a reminder
public void greetings()
{
    System.out.println(this + " says hello");
}
*****/
```

Inheritance 5-10

Printing objects



VerbosePoint.java



VerbosePoint.class

```
class VerbosePoint extends I
{
    public int x;
    public int y;

    public VerbosePoint(int x_coord, int y_coord)
    {
        this.x = x_coord; this.y = y_coord;
    }
    public VerbosePoint()
    {
        x = 0; y = 0;
    }

    public void greetings()
    {
        System.out.println(this + "says hello");
    }

    public String toString ()
    {
        return getClass().getName() + " at " + x + "," + y;
    }
}
```

Inheritance 5-11

ColorPoints



ColorPoint.java



ColorPoint.class

```
class ColorPoint extends VerbosePoint
{
    public int x;
    public int y;
    public Color color; // New instance variable

    public ColorPoint(int x_coord, int y_coord, Color c_)
    {
        x = x_coord; y = y_coord;
        this.color = c_;
    }

    public ColorPoint()
    {
        x = 0; y = 0;
        color = Color.red;
    }

    public String toString ()
    {
        return getClass().getName() + " at " + x + "," + y
            + " colored " + color;
    }
}
```

Inheritance 5-12

Testing ColorPoints



TestColorPoint.java



TestColorPoint.class

```
import java.awt.Color;

// Class used to test ColorPoint objects
public class TestColorPoint
{
    public static void main (String [] args)
    {
        ColorPoint p1 = new ColorPoint(5, 11, Color.green);
        ColorPoint p2 = new ColorPoint();

        p1.greetings();
        p2.greetings();
    }
}
```

Inheritance 5-13

Class variables and methods

```
class ColorPoint extends VerbosePoint
{
    public int x;
    public int y;
    public Color color;
    public static int numPoints = 0; // num of points created

    public ColorPoint(int x_coord, int y_coord, Color c_)
    {
        x = x_coord; y = y_coord;
        this.color = c_;
        numPoints++;
    }

    public ColorPoint()
    {
        x = 0; y = 0;
        color = c_;
        numPoints++;
    }

    public static int count() // returns num points created
    {
        return numPoints;
    }
    ... // rest same as before
}
```

Inheritance 5-14

Testing our class variables and methods

```
import java.awt.Color;

// Class used to test Point objects
public class TestColorPoint
{
    public static void main (String [] args)
    {
        ColorPoint p1 = new ColorPoint(5, 11, Color.green);
        ColorPoint p2 = new ColorPoint();

        System.out.println("You have created " +
                           ColorPoint.count() + " points");
    }
}
```

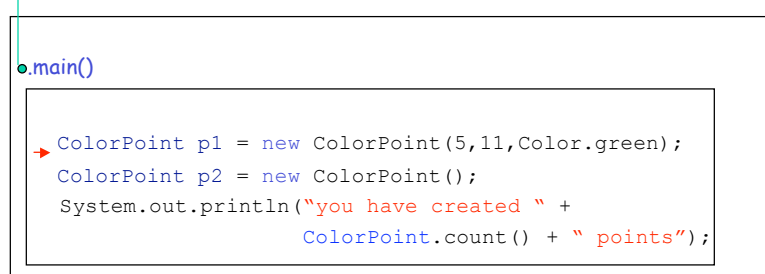
Inheritance 5-15

The `main()` method is invoked

Object Land



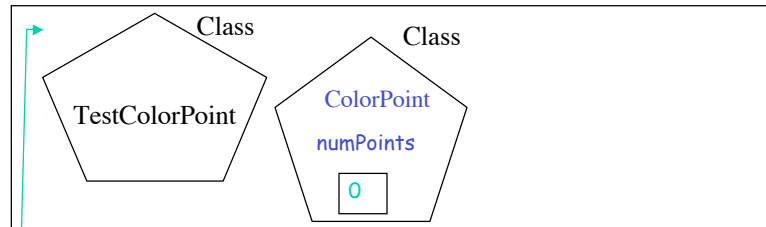
Execution Land



Inheritance 5-16

Static variables are created

Object Land



Execution Land

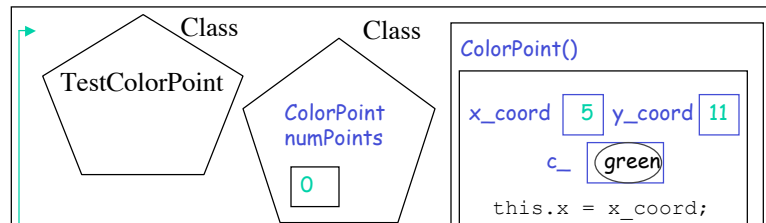
• .main()

```
ColorPoint p1 = new ColorPoint(5,11,Color.green);  
ColorPoint p2 = new ColorPoint();  
System.out.println("you have created " +  
    ColorPoint.count() + " points");
```

Inheritance 5-17

new ColorPoint() is invoked

Object Land



Execution Land

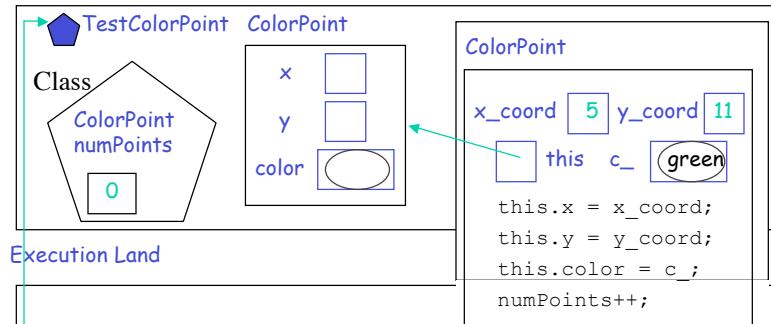
• .main()

```
ColorPoint p1 = new ColorPoint(5,11,Color.green);  
ColorPoint p2 = new ColorPoint();  
System.out.println("you have created " +  
    ColorPoint.count() + " points");
```

Inheritance 5-18

"new" does its magic

Object Land



Execution Land

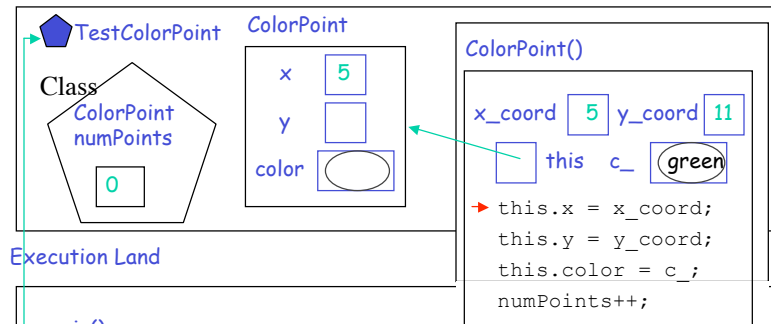
.main()

```
ColorPoint p1 = new ColorPoint(5,11,Color.green);  
ColorPoint p2 = new ColorPoint();  
System.out.println("you have created " +  
    ColorPoint.count() + " points");
```

Inheritance 5-19

And ColorPoint() executes

Object Land



Execution Land

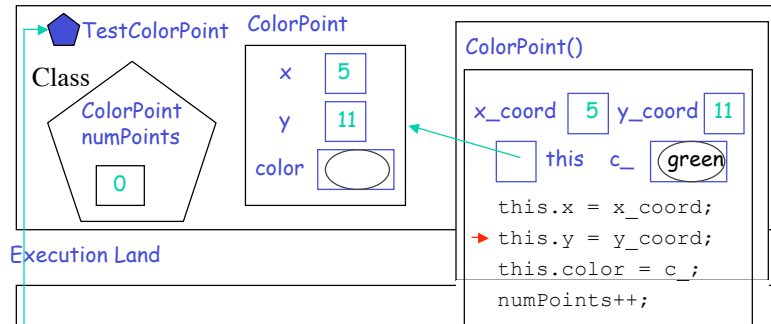
.main()

```
ColorPoint p1 = new ColorPoint(5,11,Color.green);  
ColorPoint p2 = new ColorPoint();  
System.out.println("you have created " +  
    ColorPoint.count() + " points");
```

Inheritance 5-20

Instance variables are filled

Object Land



Execution Land

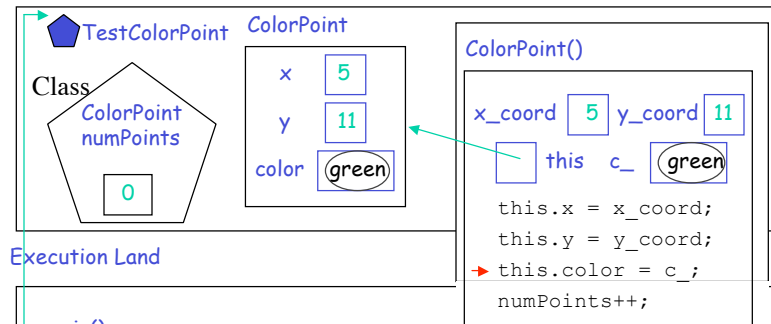
• `.main()`

```
ColorPoint p1 = new ColorPoint(5,11,Color.green);
ColorPoint p2 = new ColorPoint();
System.out.println("you have created " +
    ColorPoint.count() + " points");
```

Inheritance 5-21

color is assigned

Object Land



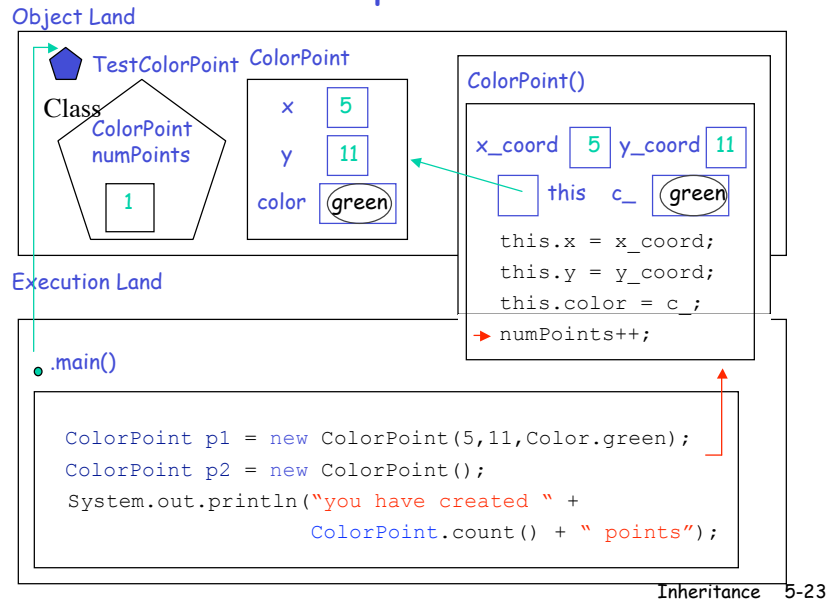
Execution Land

• `.main()`

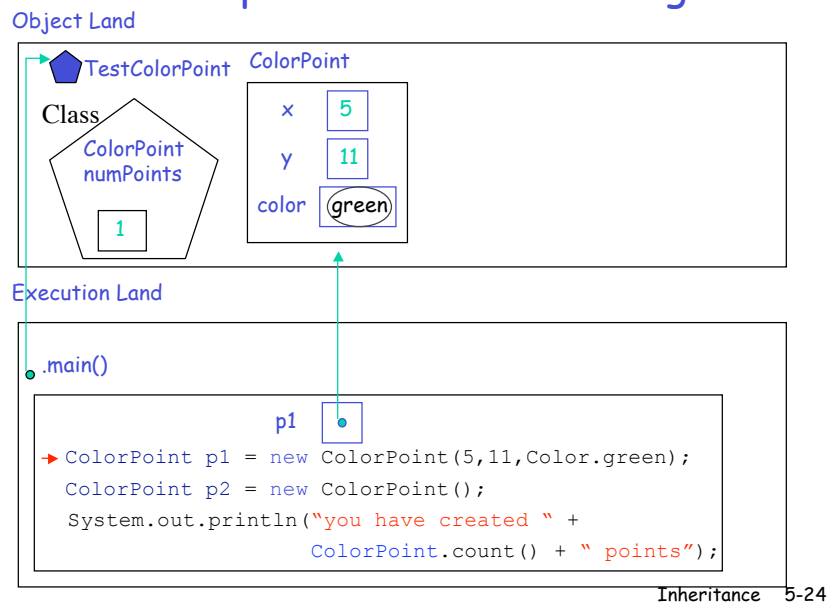
```
ColorPoint p1 = new ColorPoint(5,11,Color.green);
ColorPoint p2 = new ColorPoint();
System.out.println("you have created " +
    ColorPoint.count() + " points");
```

Inheritance 5-22

Static numPoints is updated

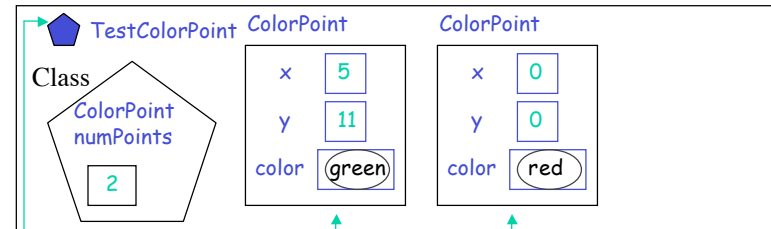


Local variable p1 is created and assigned



A second Point is born

Object Land



Execution Land

• `.main()`

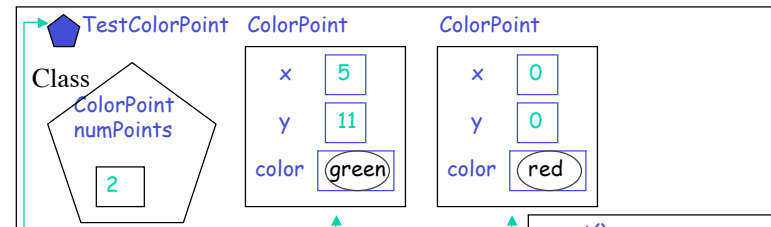
```

    p1
    ColorPoint p1 = new ColorPoint(5,11,Color.green);
    ColorPoint p2 = new ColorPoint();
    System.out.println("you have created " +
        ColorPoint.count() + " points");
    
```

Inheritance 5-25

Greetings are sent

Object Land



Execution Land

• `.main()`

```

    p1
    ColorPoint p1 = new ColorPoint(5,11,Color.green);
    ColorPoint p2 = new ColorPoint();
    System.out.println("you have created " +
        ColorPoint.count() + " points");
    
```

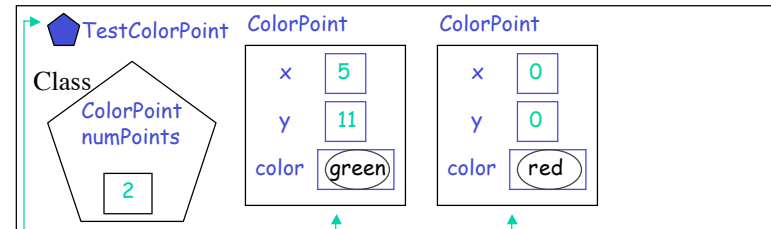
```

count()
return numPoints;
    
```

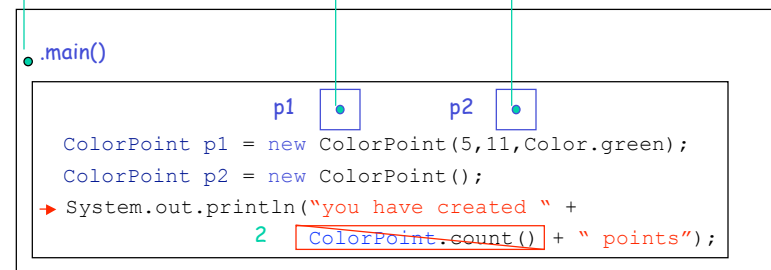
Inheritance 5-26

count(), System.out.println() finish

Object Land



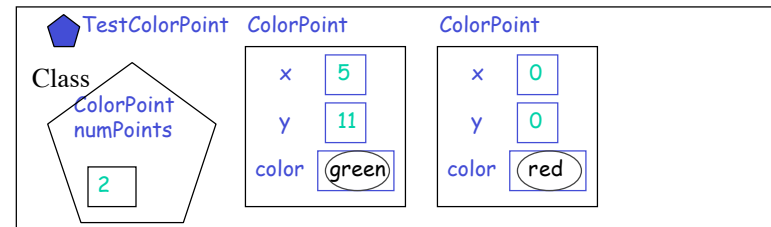
Execution Land



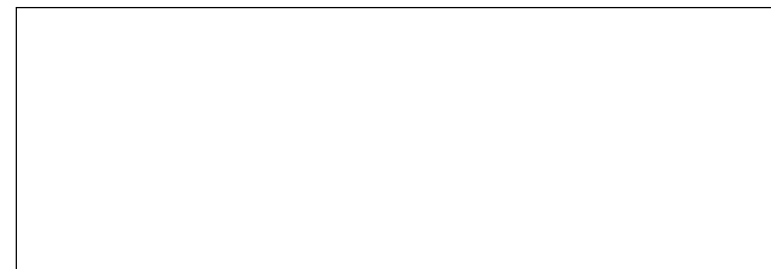
Inheritance 5-27

main closes ...

Object Land



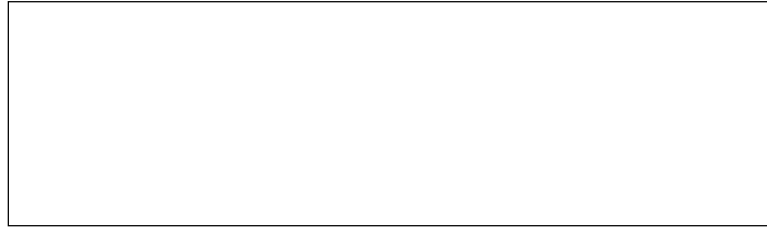
Execution Land



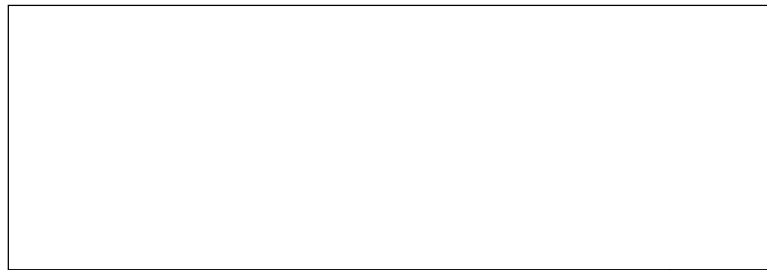
Inheritance 5-28

... and garbage collection occurs

Object Land



Execution Land



Inheritance 5-29

A class is described by

Entity

Instance variables

Class variables

Local variables

Constructor
methods

Instance
methods

Class
methods

Declaration

`public int x`

`public static final Color yellow`

never see it

`public Color(int r,int g,int b)`

`public void setColor(Color c)`

`public static int max(int a, int b)`

How to Use

`transportRoom.x`

`<exp>.<varName>`

`Color.yellow`

`<className>.<varName>`

`p`

`<varname>`

`new Color(0,0,0);`

`betty.setColor(c);`

`Math.max(3,7);`

Inheritance 5-30