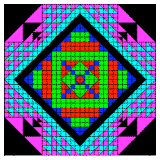
Inheritance

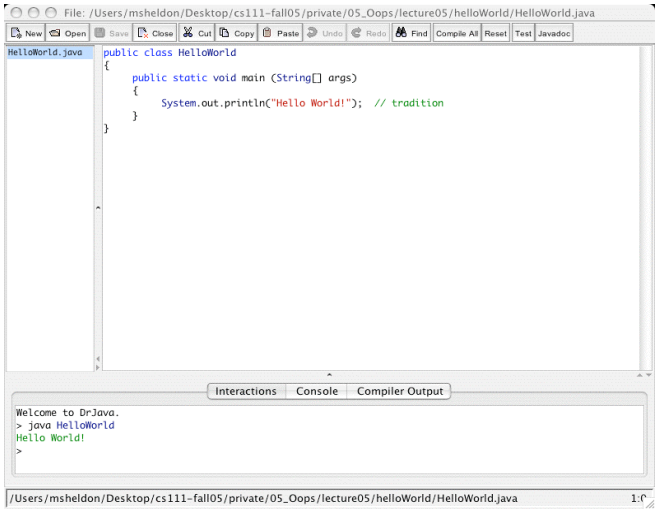
Methods of modularity



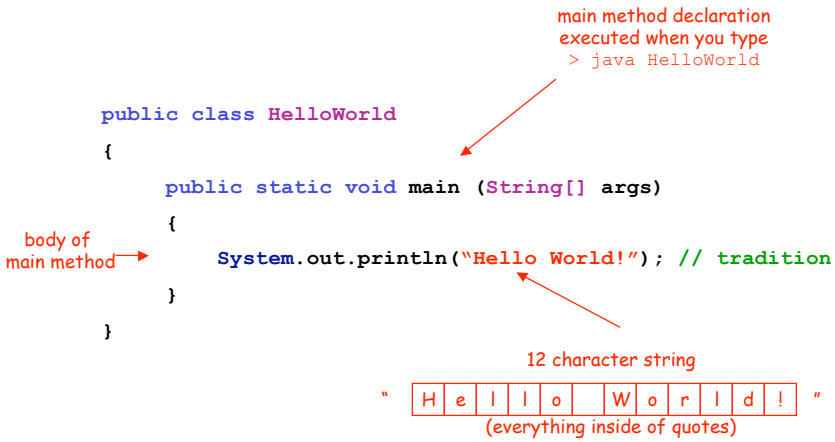
CS111 Computer Programming

Department of Computer Science
Wellesley College

Our first application



A closer look



A variation

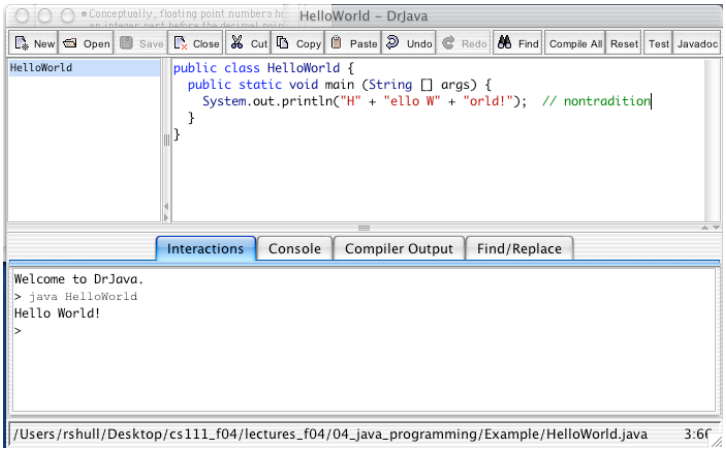
```
public class HelloWorld
{
     public static void main (String[] args)
     {
          System.out.print("Hello "); // also tradition
          System.out.println("World!");
     }
}
```

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.

String concatenation*



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

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"};
     }
}
```


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);
     }
}
```

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");
     }
}
```


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");
     }
*****************************************/
```

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;
     }
}
```


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;
     }
}
```

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();
     }
}
```

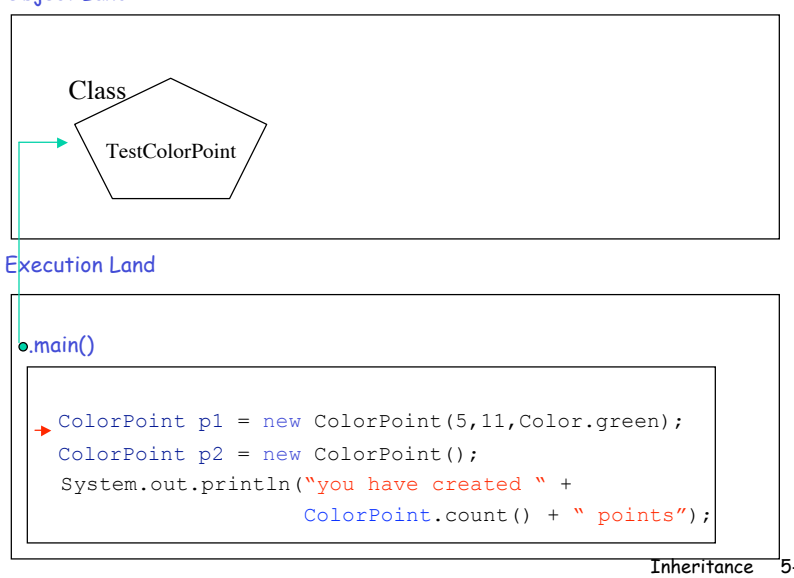

Class variables and methods

```
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
}
```

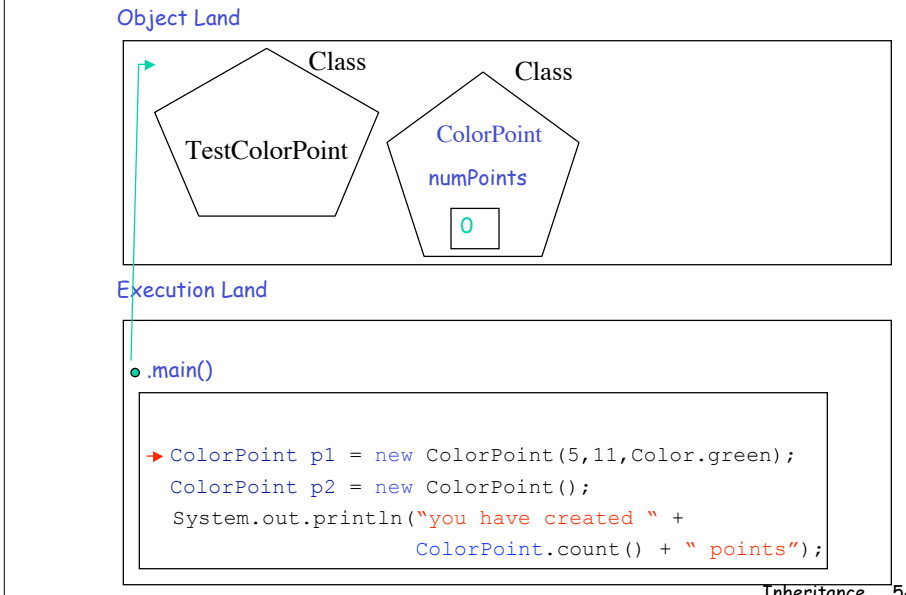
Testing our class variables and methods

```
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");
     }
}
```

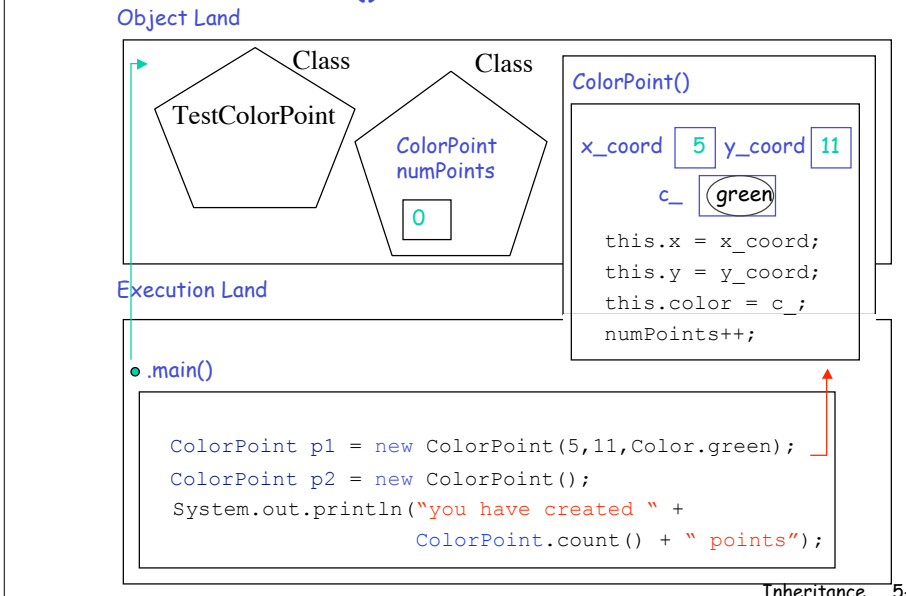

The `main()` method is invoked



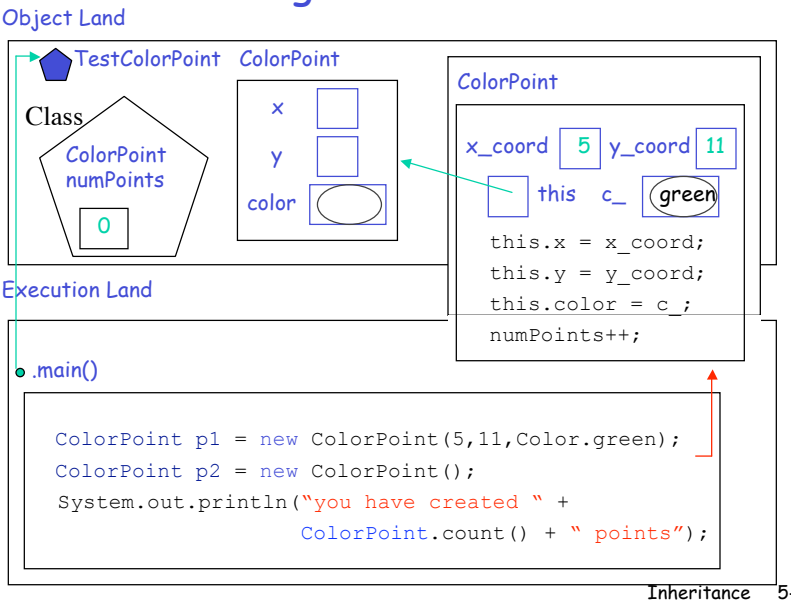
Static variables are created



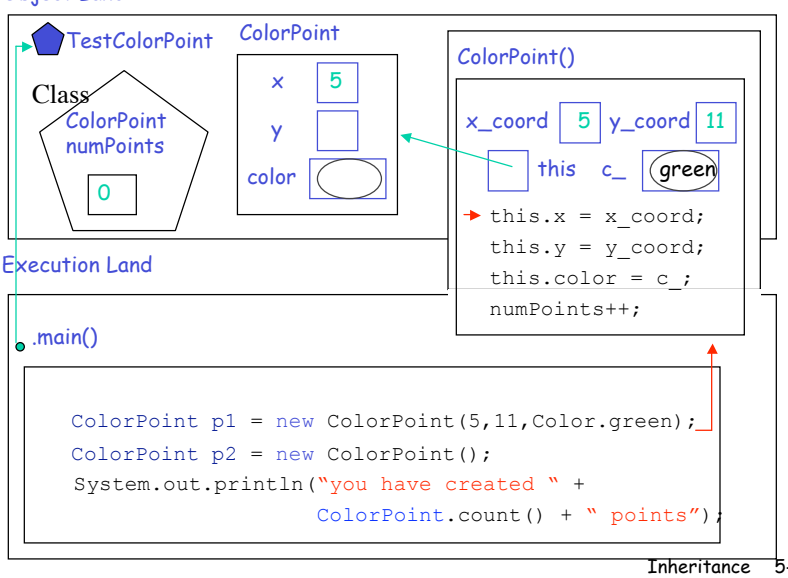
new ColorPoint() is invoked



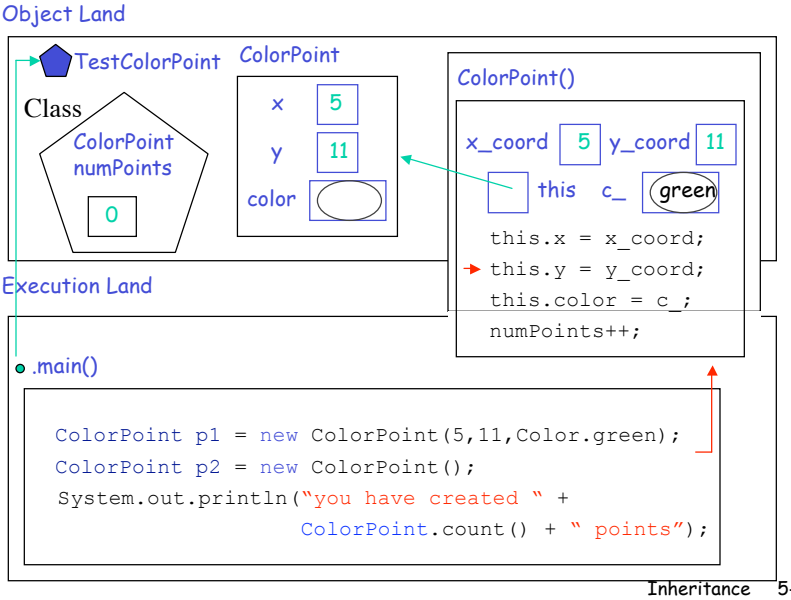
"new" does its magic



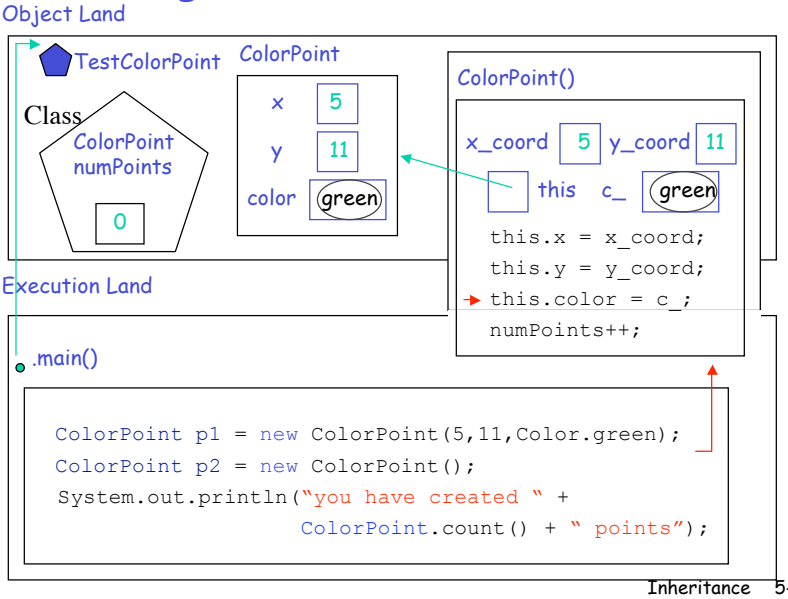
And ColorPoint() executes



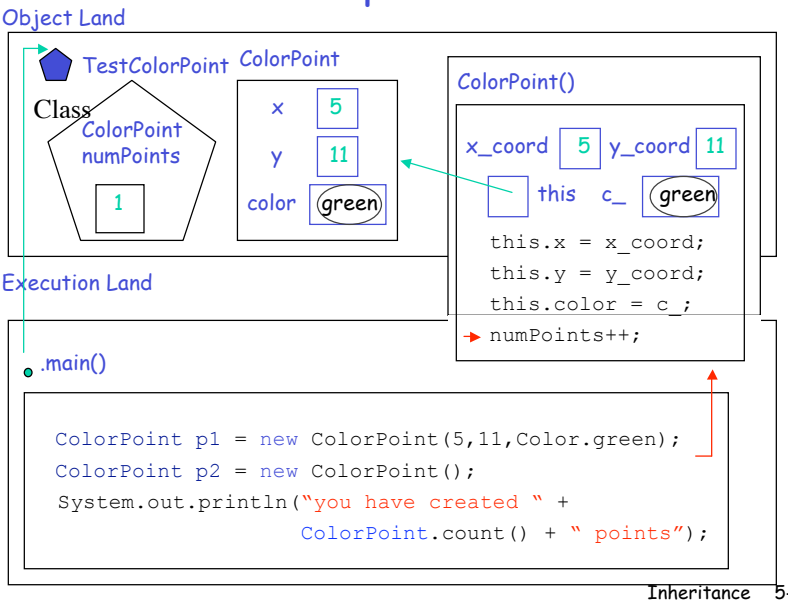
Instance variables are filled



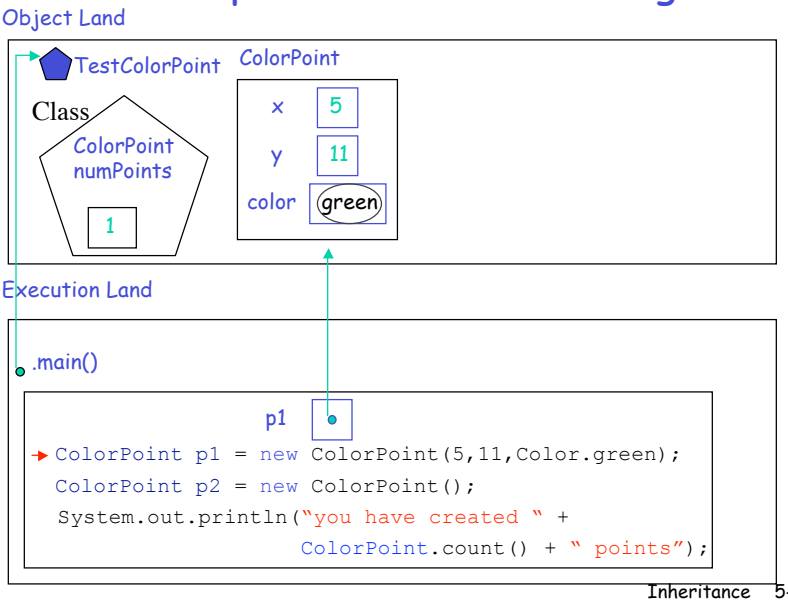
color is assigned



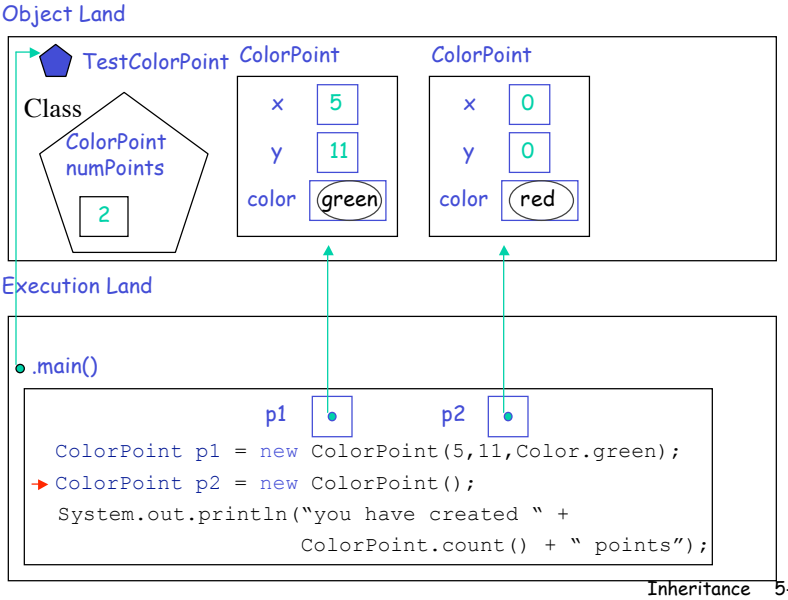
Static numPoints is updated



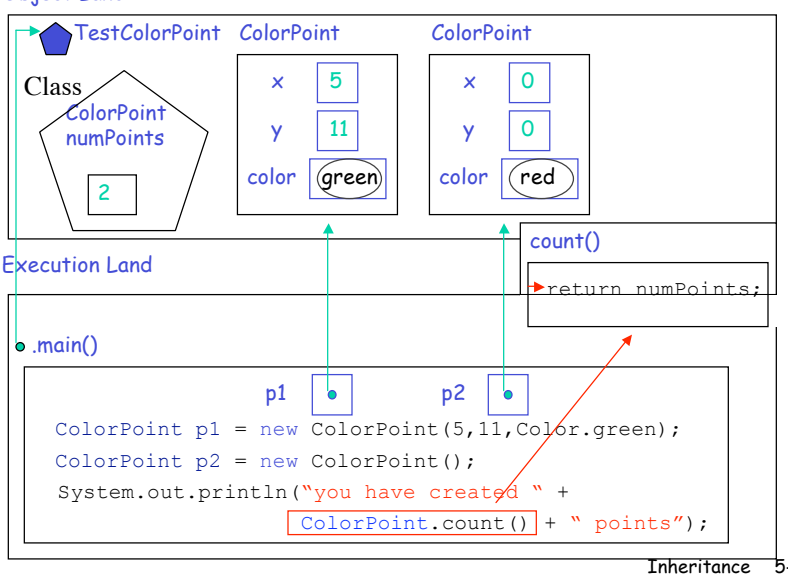
Local variable p1 is created and assigned



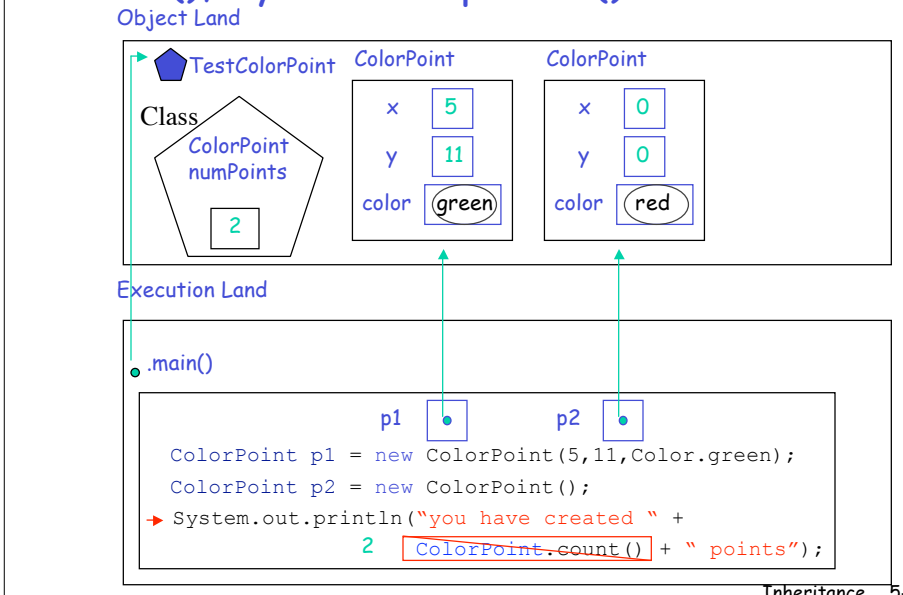
A second Point is born



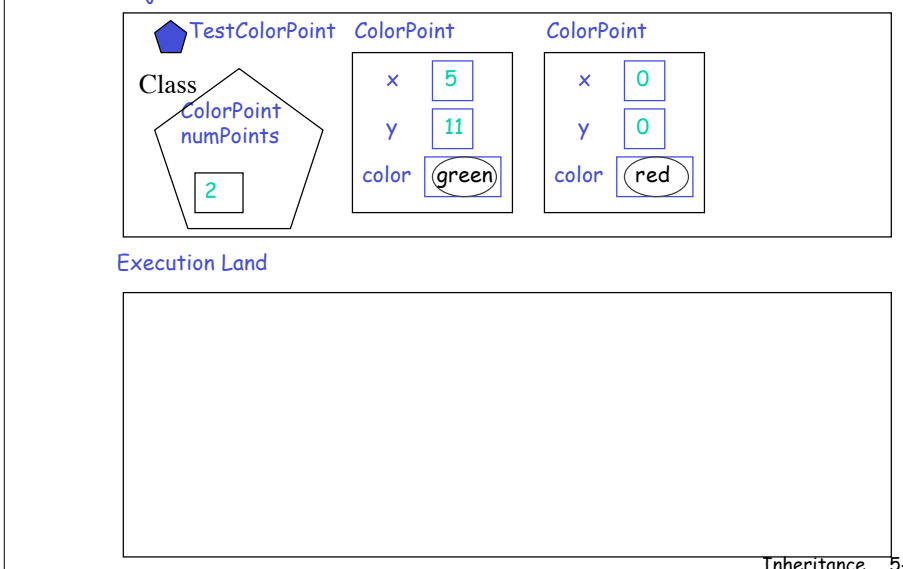
Greetings are sent



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

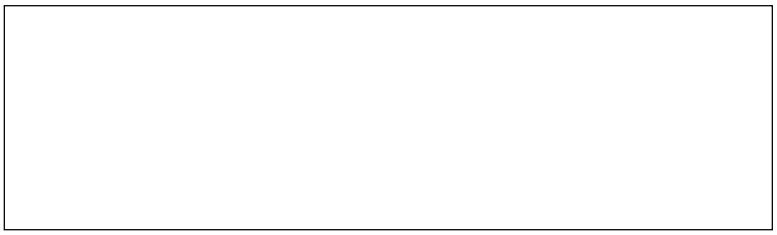


main closes . . .

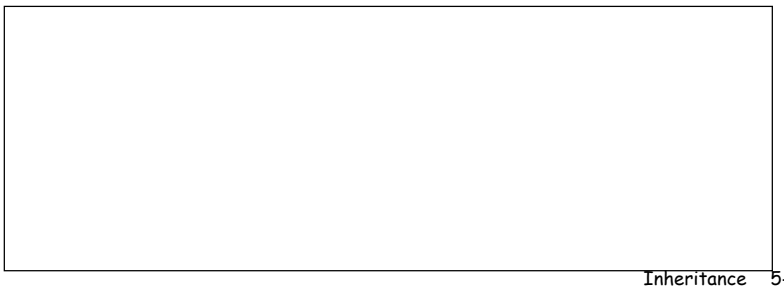


. . . and garbage collection occurs

Object Land



Execution Land



A class is described by

Entity	Declaration	How to Use
Instance variables	`public int x`	`transportRoom.x` <exp>.<varName>
Class variables	`public static final Color yellow`	`Color.yellow` <className>.<varName>
Local variables	never see it	p <varname>
Constructor methods	`public Color(int r,int g,int b)`	`new Color(0,0,0);`
Instance methods	`public void setColor(Color c)`	`betty.setColor(c);`
Class methods	`public static int max(int a, int b)`	`Math.max(3,7);`