GEOG 178/258
Week 5:

Overriding, Overloading, Inheritance

mike johnson

If you are feeling behind,

- Download the sample code from the class site:
  - It contains a completed class for:
    - Point
    - Polyline
    - Polygon
    - Person

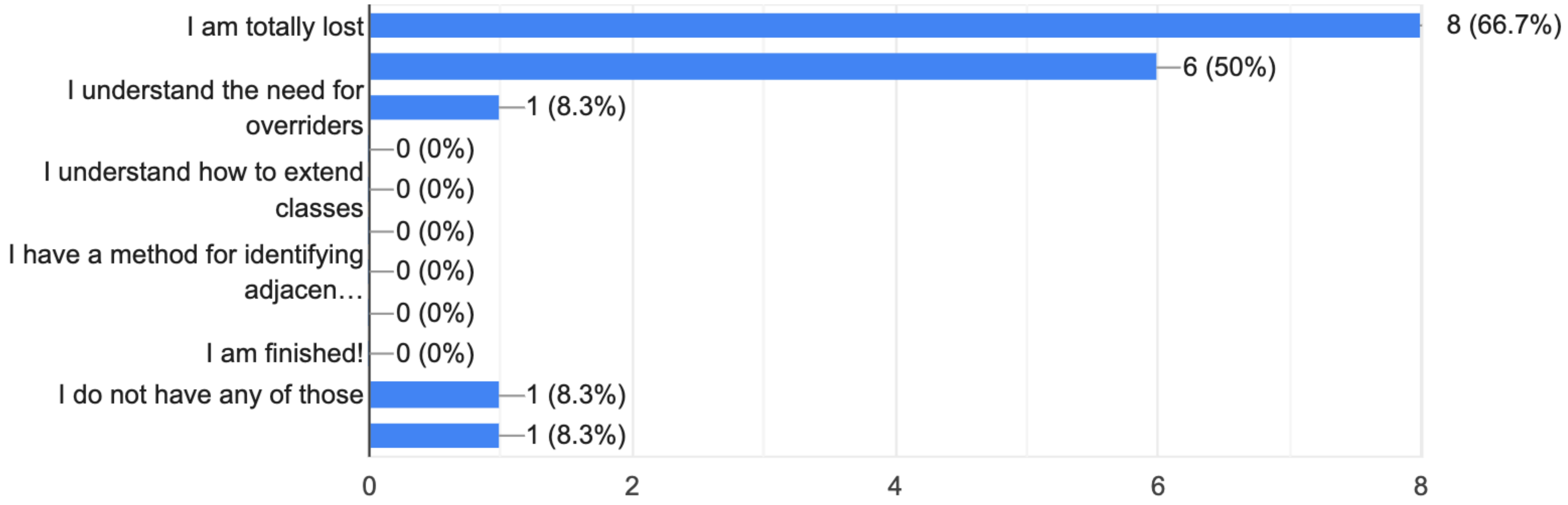
Set up:

Before we get started lets set up for this weeks lab:

Create a new project (week5) and copy over your:

Point Class	Bbox Class	Person Class

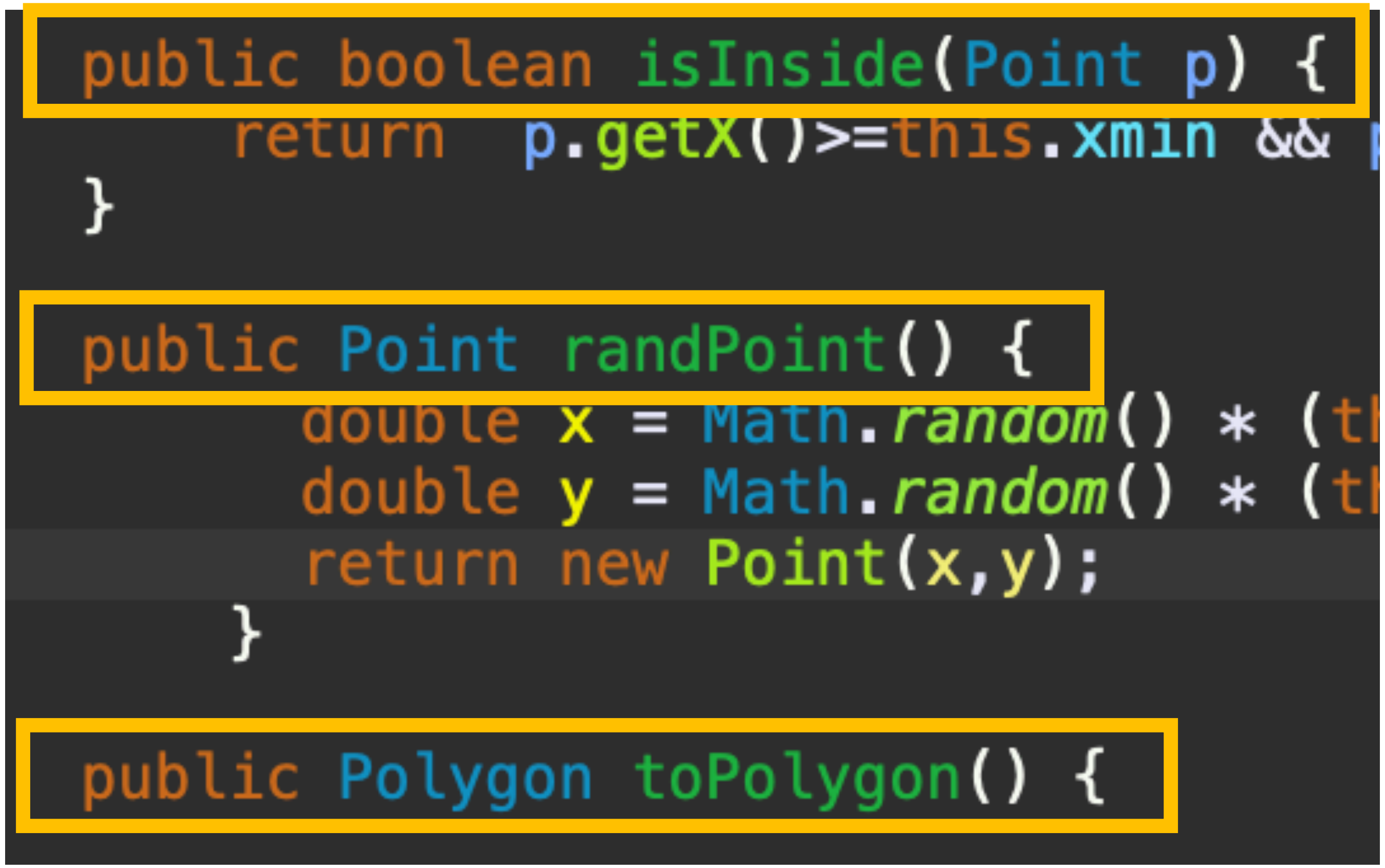
12 responses



You are not alone!!

Recap

- Method Signatures
- Constructor Signatures
- Delegation

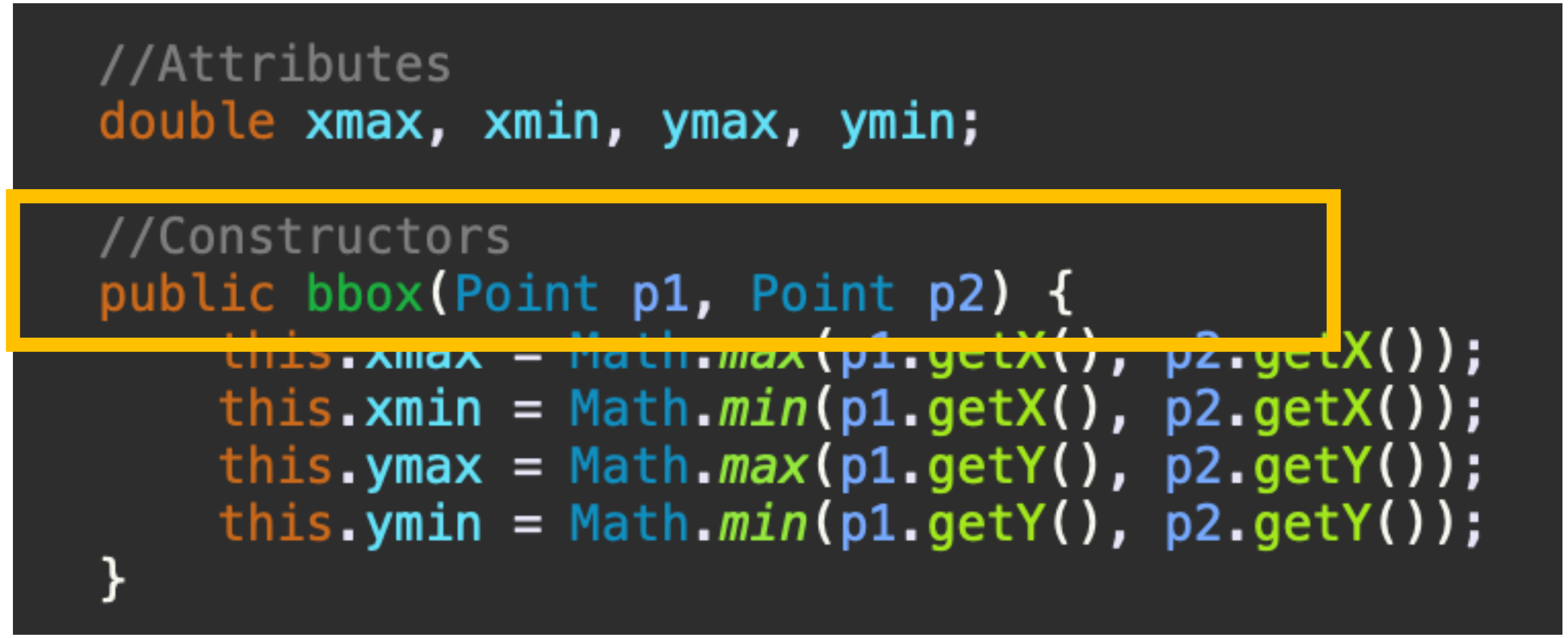


```
public boolean isInside(Point p) {
        return  p.getX()>=this.xmin && 
}

public Point randPoint() {
        double x = Math.random() * (t
        double y = Math.random() * (t
        return new Point(x,y);
    }

public Polygon toPolygon() {
```

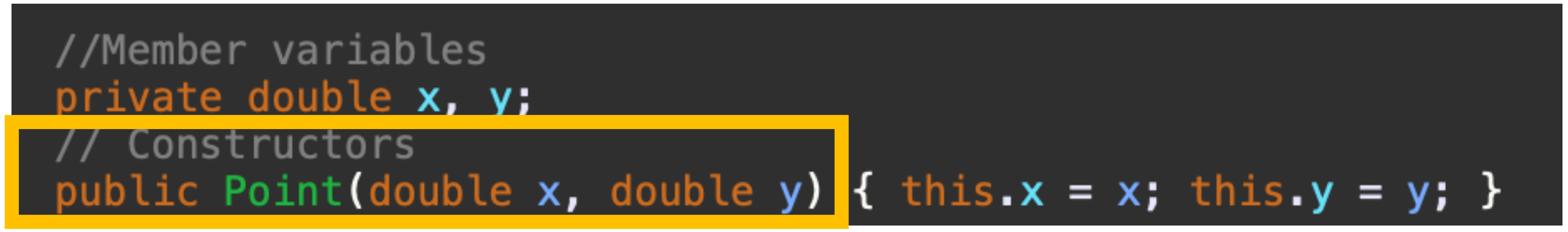
Visibility - return type – name -   inputs

Recap

- Method Signatures
- Constructor Signatures
- Delegation



```
//Attributes
double xmax, xmin, ymax, ymin;

//Constructors
public bbox(Point p1, Point p2) {
    this.xmax = Math.max(p1.getX(), p2.getX());
    this.xmin = Math.min(p1.getX(), p2.getX());
    this.ymax = Math.max(p1.getY(), p2.getY());
    this.ymin = Math.min(p1.getY(), p2.getY());
}
```

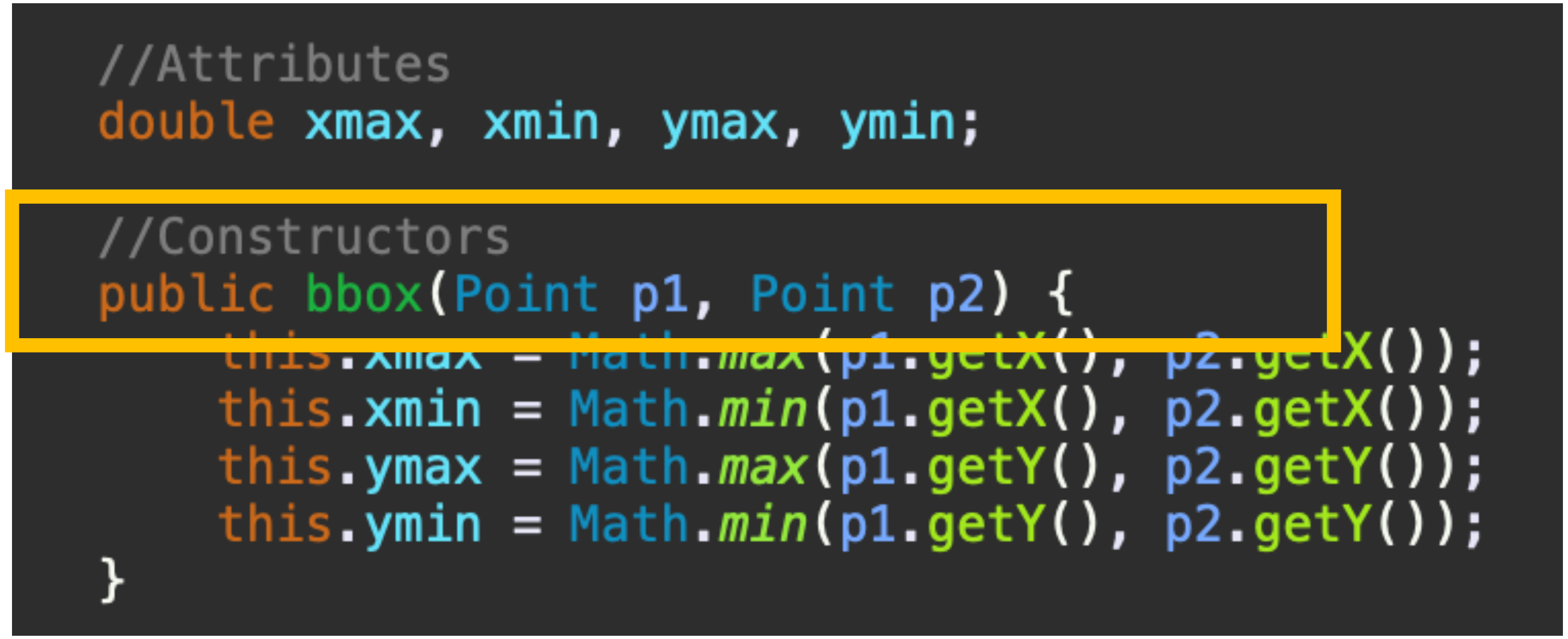


```
//Member variables
private double x, y;
// Constructors
public Point(double x, double y) { this.x = x; this.y = y; }
```

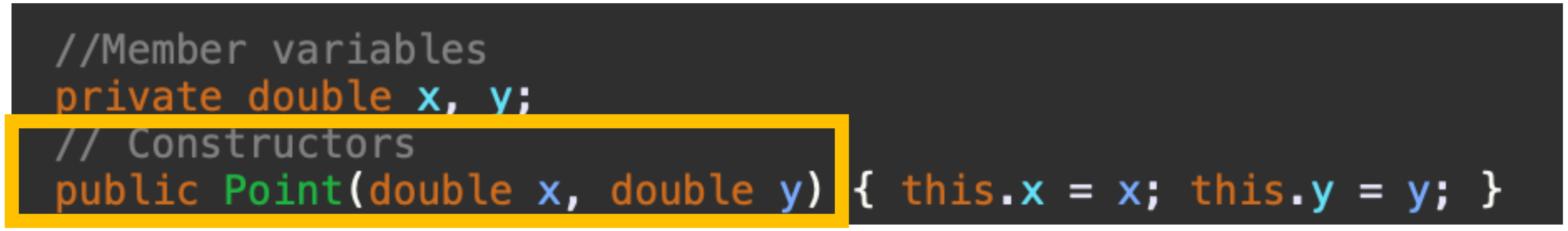
Visibility – Name that matches class -- Input

Recap

- Method Signatures
- Constructor Signatures
- Delegation



```
//Attributes
double xmax, xmin, ymax, ymin;

//Constructors
public bbox(Point p1, Point p2) {
    this.xmax = Math.max(p1.getX(), p2.getX());
    this.xmin = Math.min(p1.getX(), p2.getX());
    this.ymax = Math.max(p1.getY(), p2.getY());
    this.ymin = Math.min(p1.getY(), p2.getY());
}
```

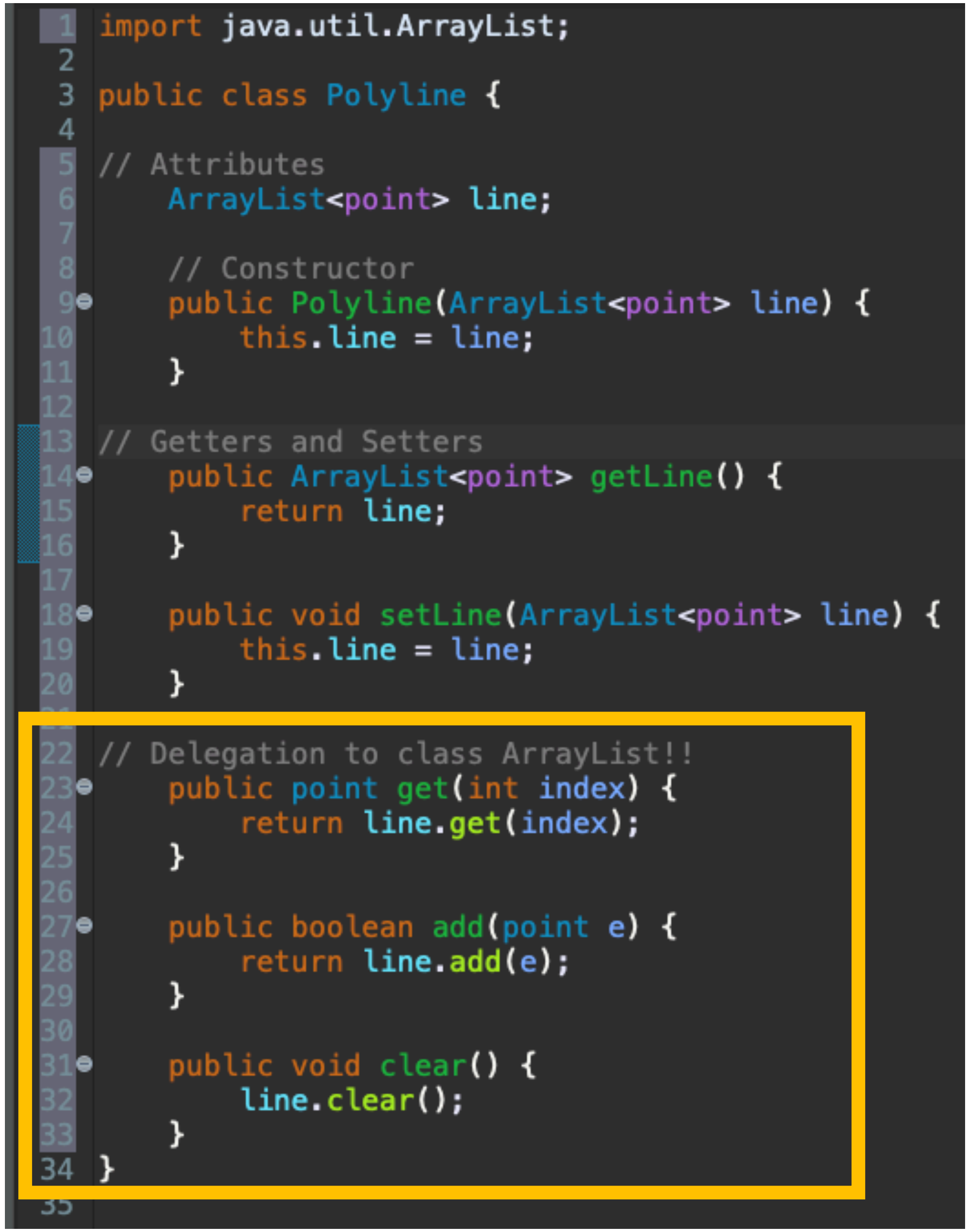


```
//Member variables
private double x, y;
// Constructors
public Point(double x, double y) { this.x = x; this.y = y; }
```

Visibility – Name that matches class -- Input

Delegation

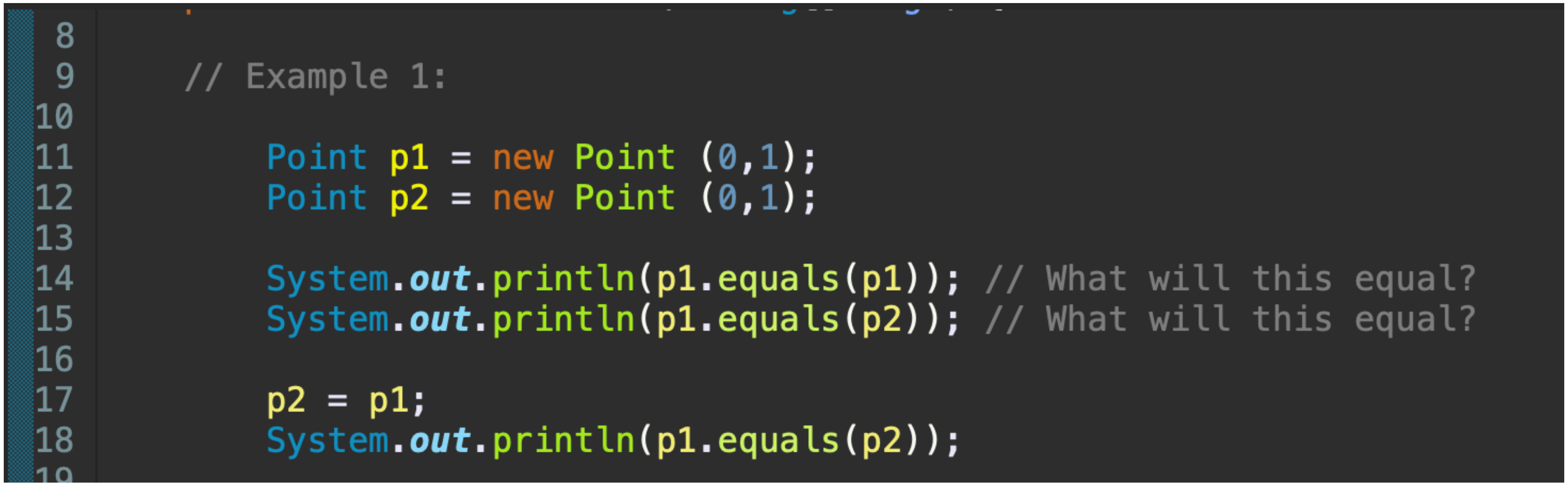
- Passing your work (a duty) over to someone/something else.
- When you delegate, **you are simply calling up some class which knows what must be done**. You do not really care how it does it, all you care about is that the class you are calling knows what needs doing.



```java
import java.util.ArrayList;

public class Polyline {

// Attributes
    ArrayList<point> line;

    // Constructor
    public Polyline(ArrayList<point> line) {
        this.line = line;
    }

// Getters and Setters
    public ArrayList<point> getLine() {
        return line;
    }

    public void setLine(ArrayList<point> line) {
        this.line = line;
    }

// Delegation to class ArrayList!!
    public point get(int index) {
        return line.get(index);
    }

    public boolean add(point e) {
        return line.add(e);
    }

    public void clear() {
        line.clear();
    }
}
```

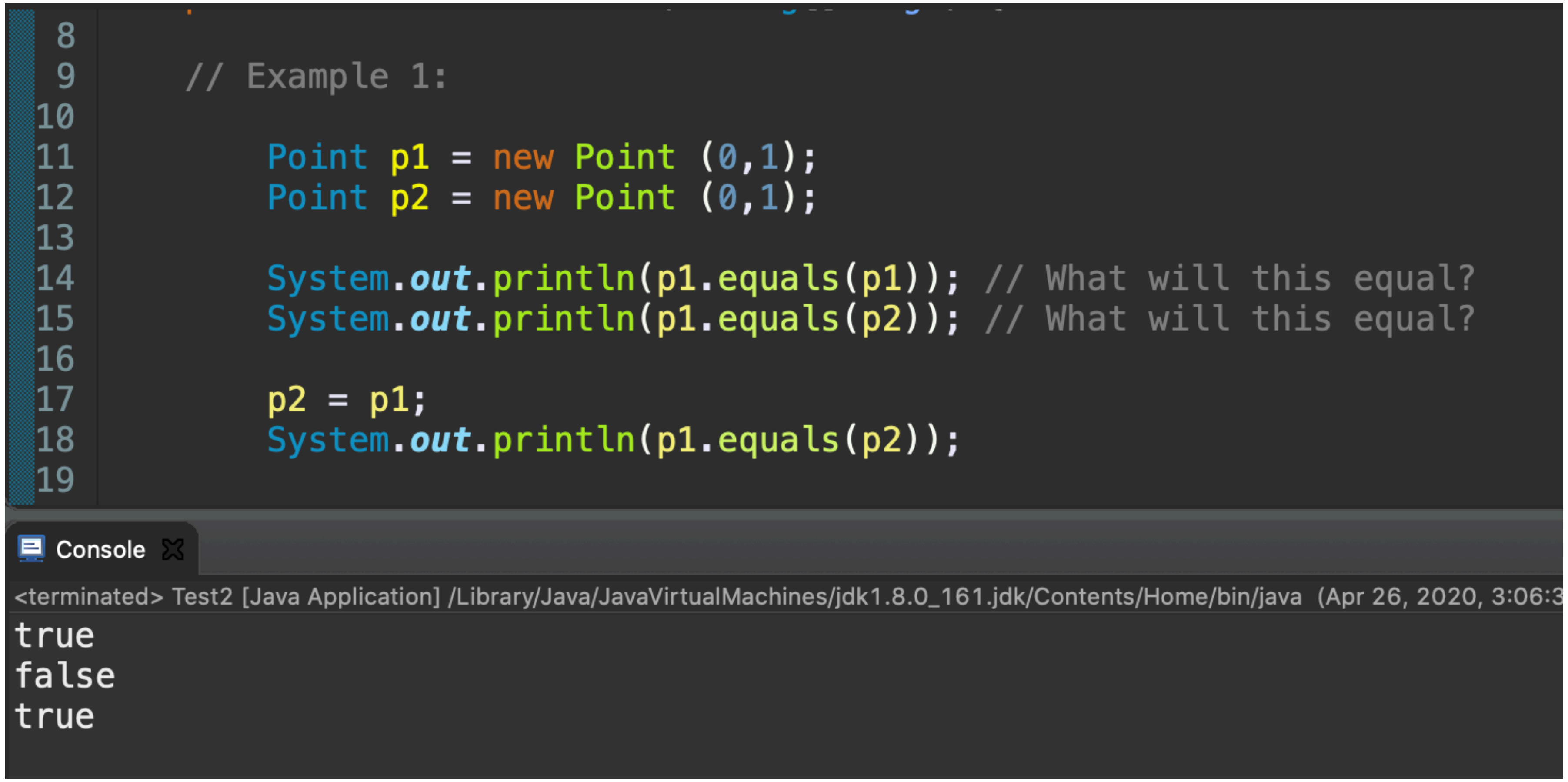
@Overriding

Example 1: Example 1: <class>.**equals()** <class>.**euals()**



```
    // Example 1:

        Point p1 = new Point (0,1);
        Point p2 = new Point (0,1);

        System.out.println(p1.equals(p1)); // What will this equal?
        System.out.println(p1.equals(p2)); // What will this equal?

        p2 = p1;
        System.out.println(p1.equals(p2));
```

Example 1: <class>.**equals()**

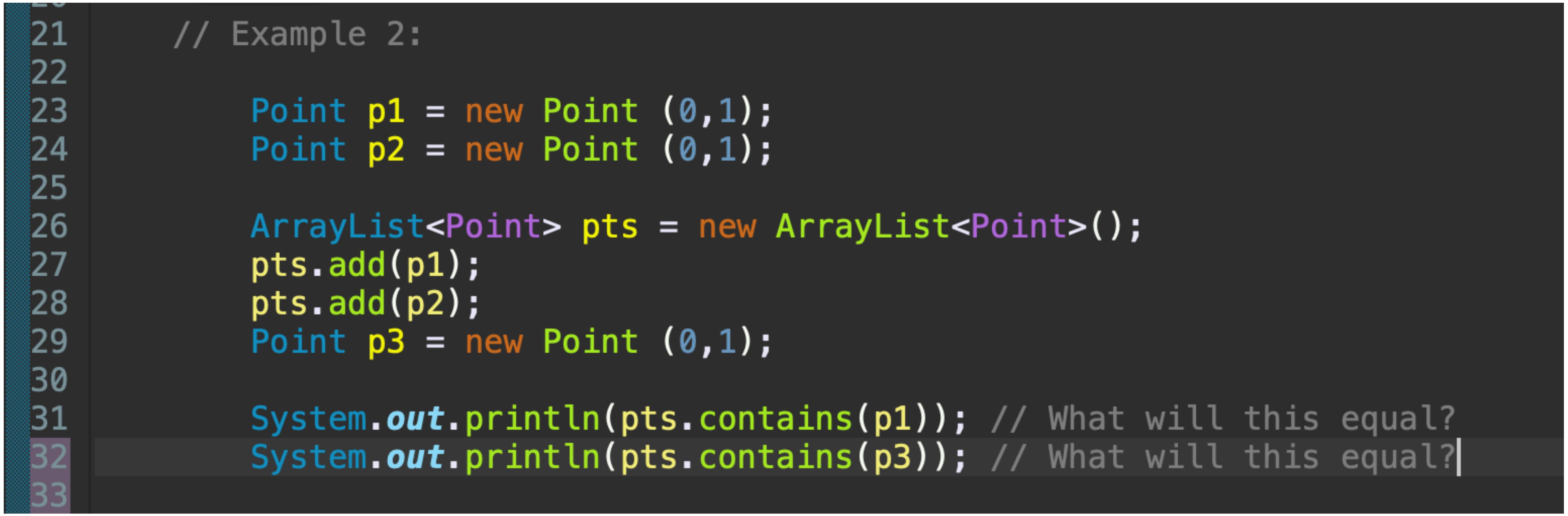


```
    // Example 1:

        Point p1 = new Point (0,1);
        Point p2 = new Point (0,1);

        System.out.println(p1.equals(p1)); // What will this equal?
        System.out.println(p1.equals(p2)); // What will this equal?

        p2 = p1;
        System.out.println(p1.equals(p2));
```

Console

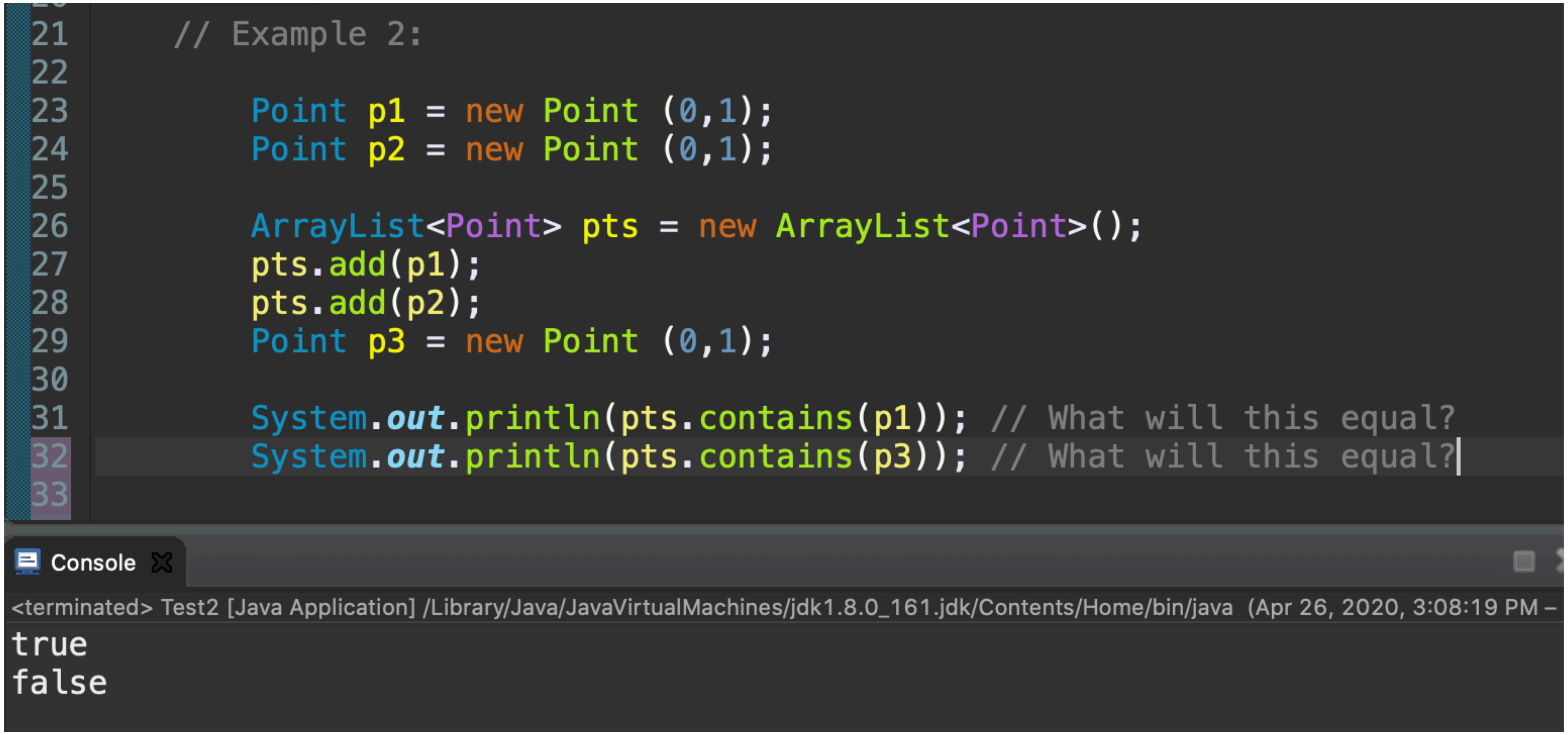
```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 26, 2020, 3:06:3
true
false
true
```

Example 2: ArrayList<>().contains()



```java
    // Example 2:

        Point p1 = new Point (0,1);
        Point p2 = new Point (0,1);

        ArrayList<Point> pts = new ArrayList<Point>();
        pts.add(p1);
        pts.add(p2);
        Point p3 = new Point (0,1);

        System.out.println(pts.contains(p1)); // What will this equal?
        System.out.println(pts.contains(p3)); // What will this equal?
```

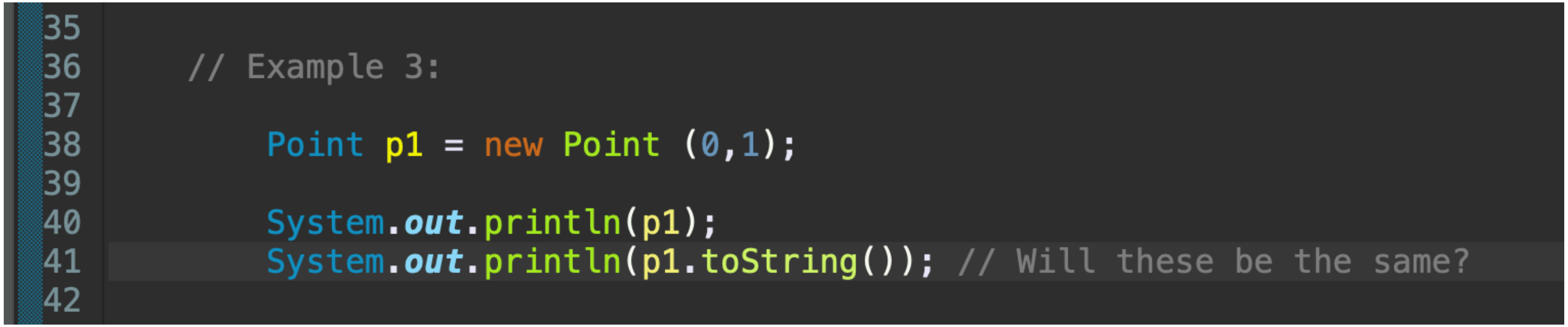
Example 2: ArrayList<>().contains()



```
    // Example 2:

        Point p1 = new Point (0,1);
        Point p2 = new Point (0,1);

        ArrayList<Point> pts = new ArrayList<Point>();
        pts.add(p1);
        pts.add(p2);
        Point p3 = new Point (0,1);

        System.out.println(pts.contains(p1)); // What will this equal?
        System.out.println(pts.contains(p3)); // What will this equal?
```

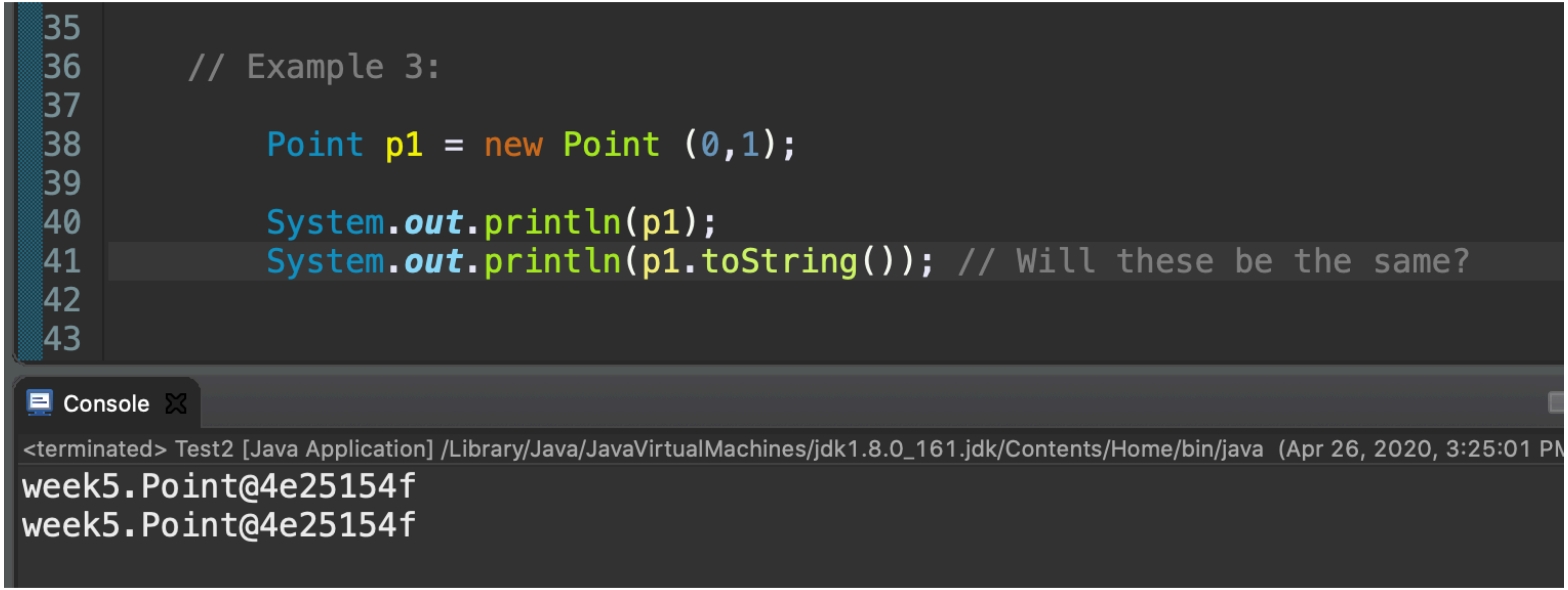
```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 26, 2020, 3:08:19 PM –
true
false
```

Example 3: <class>.toString()



```

    // Example 3:

        Point p1 = new Point (0,1);

        System.out.println(p1);
        System.out.println(p1.toString()); // Will these be the same?

```

Example 3: <class>.toString()



```
    // Example 3:

        Point p1 = new Point (0,1);

        System.out.println(p1);
        System.out.println(p1.toString()); // Will these be the same?


```

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 26, 2020, 3:25:01 PM
week5.Point@4e25154f
week5.Point@4e25154f
```

Overriding

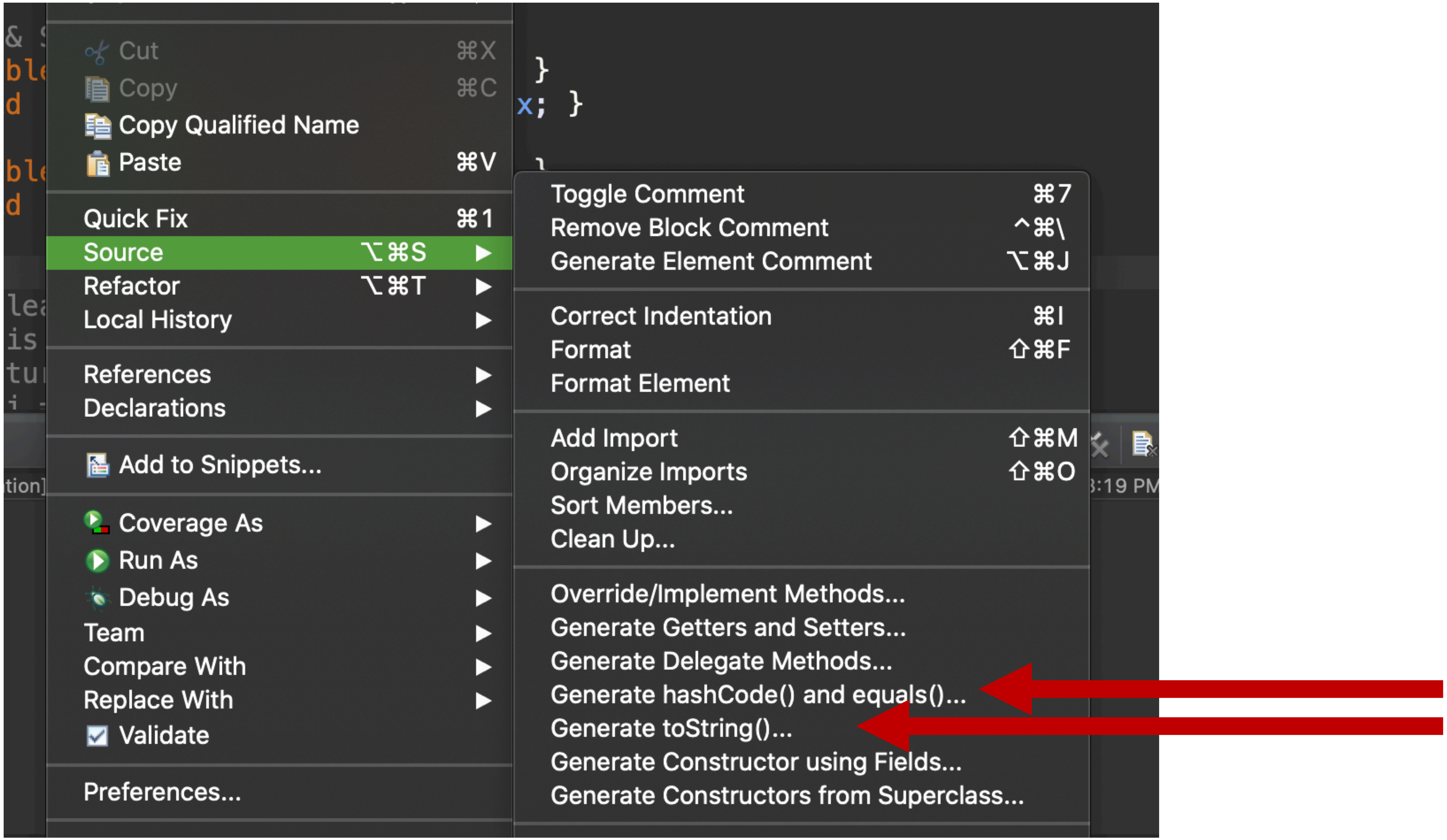
In any object-oriented programming language…

Overriding is a feature

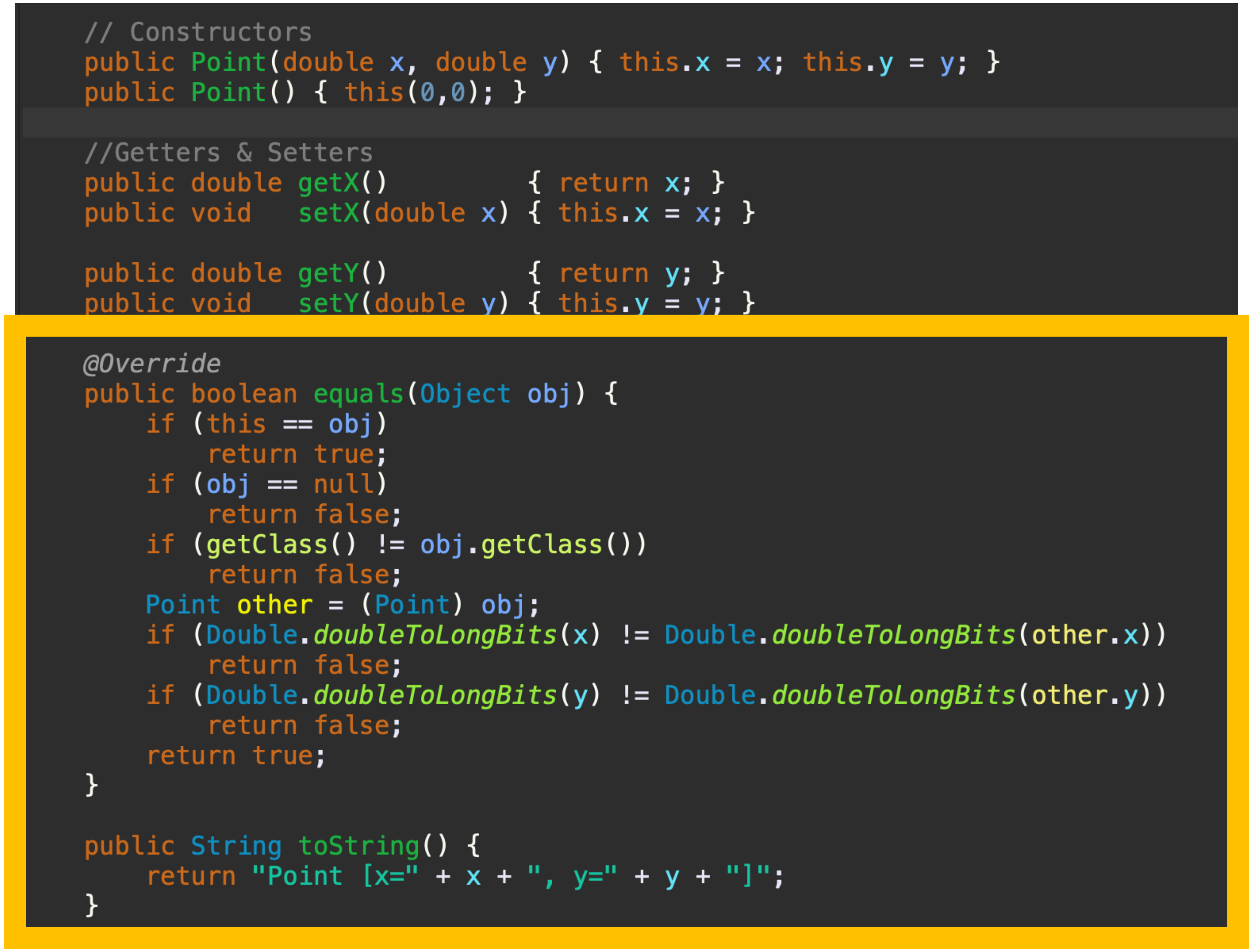
that allows a subclass or child class to provide

a unique implementation of a method that is already provided

by one of its super-classes or parent classes.


Cut ⌘X
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Quick Fix ⌘1
Source ⌥⌘S
Refactor ⌥⌘T
Local History
References
Declarations
Add to Snippets...
Coverage As
Run As
Debug As
Team
Compare With
Replace With
Validate
Preferences...
Toggle Comment ⌘7
Remove Block Comment ^⌘\
Generate Element Comment ⌥⌘J
Correct Indentation ⌘I
Format ⇧⌘F
Format Element
Add Import ⇧⌘M
Organize Imports ⇧⌘O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate hashCode() and equals()...
Generate toString()...
Generate Constructor using Fields...
Generate Constructors from Superclass...

Point Class

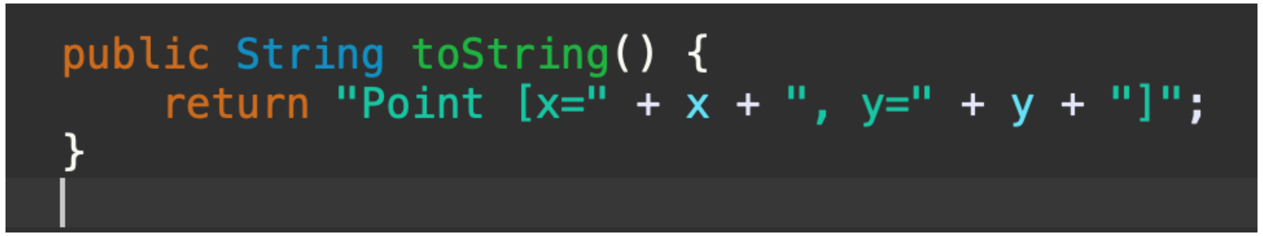


```
// Constructors
public Point(double x, double y) { this.x = x; this.y = y; }
public Point() { this(0,0); }

//Getters & Setters
public double getX()          { return x; }
public void   setX(double x) { this.x = x; }

public double getY()          { return y; }
public void   setY(double y) { this.y = y; }

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Point other = (Point) obj;
    if (Double.doubleToLongBits(x) != Double.doubleToLongBits(other.x))
        return false;
    if (Double.doubleToLongBits(y) != Double.doubleToLongBits(other.y))
        return false;
    return true;
}

public String toString() {
    return "Point [x=" + x + ", y=" + y + "]";
}
```

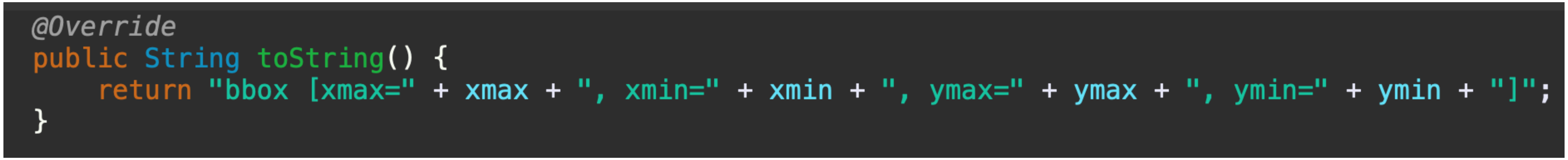
… autogenerated (no need to type out)….

Take a couple minutes to override the *toString* method for your point and bbox, and person class…

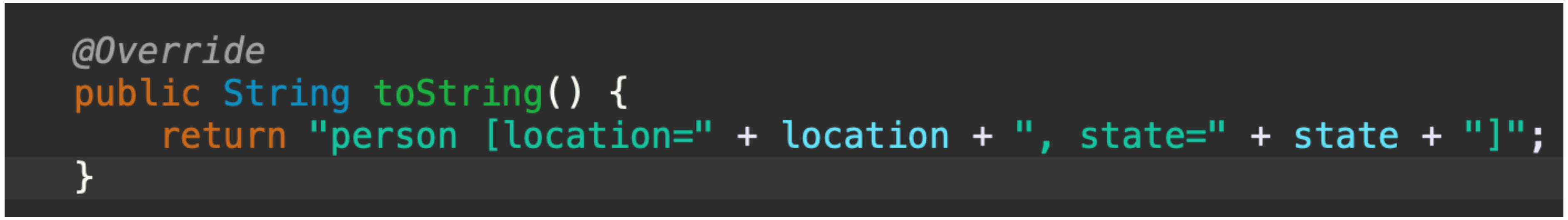
And create the equals method in your point class…



```java
public String toString() {
    return "Point [x=" + x + ", y=" + y + "]";
}
```

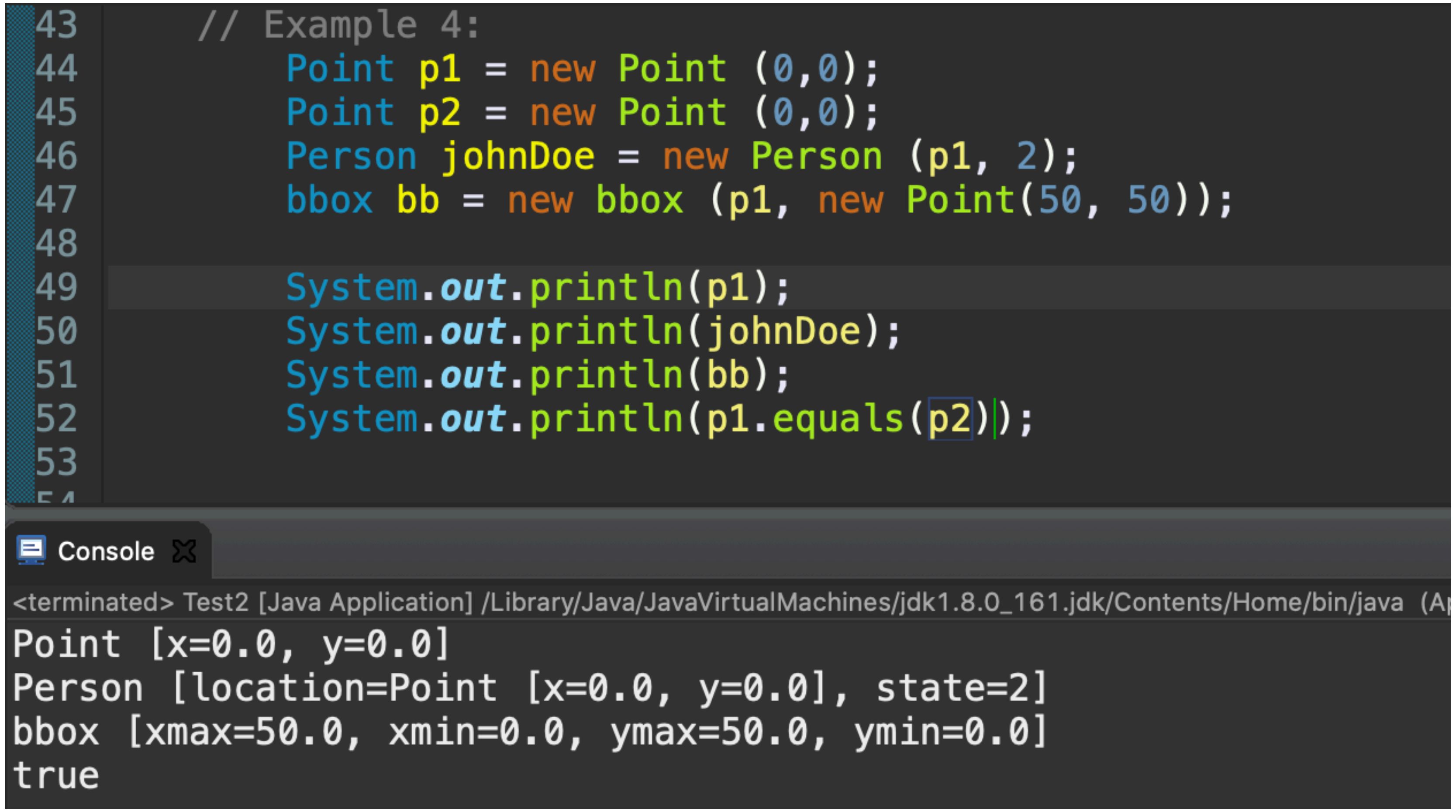


```java
@Override
public String toString() {
    return "bbox [xmax=" + xmax + ", xmin=" + xmin + ", ymax=" + ymax + ", ymin=" + ymin + "]";
}
```



```java
@Override
public String toString() {
    return "person [location=" + location + ", state=" + state + "]";
}
```

Example 4: Testing

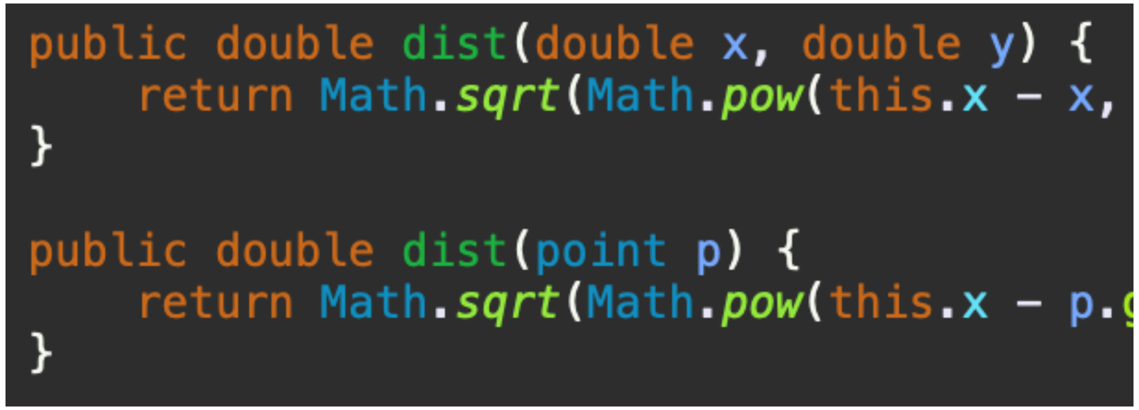


```
    // Example 4:
        Point p1 = new Point (0,0);
        Point p2 = new Point (0,0);
        Person johnDoe = new Person (p1, 2);
        bbox bb = new bbox (p1, new Point(50, 50));

        System.out.println(p1);
        System.out.println(johnDoe);
        System.out.println(bb);
        System.out.println(p1.equals(p2));
```

Console

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java (Ap
Point [x=0.0, y=0.0]
Person [location=Point [x=0.0, y=0.0], state=2]
bbox [xmax=50.0, xmin=0.0, ymax=50.0, ymin=0.0]
true
```

@Overloading



```
public double dist(double x, double y) {
    return Math.sqrt(Math.pow(this.x - x,
}

public double dist(point p) {
    return Math.sqrt(Math.pow(this.x - p.g
}
```

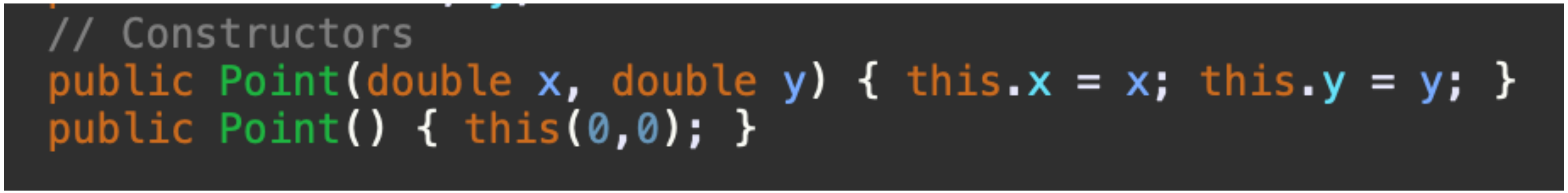
Method Overloading is a feature…

that allows a class

to have more than one method using the same name,

if their argument lists (**signatures**) are different.

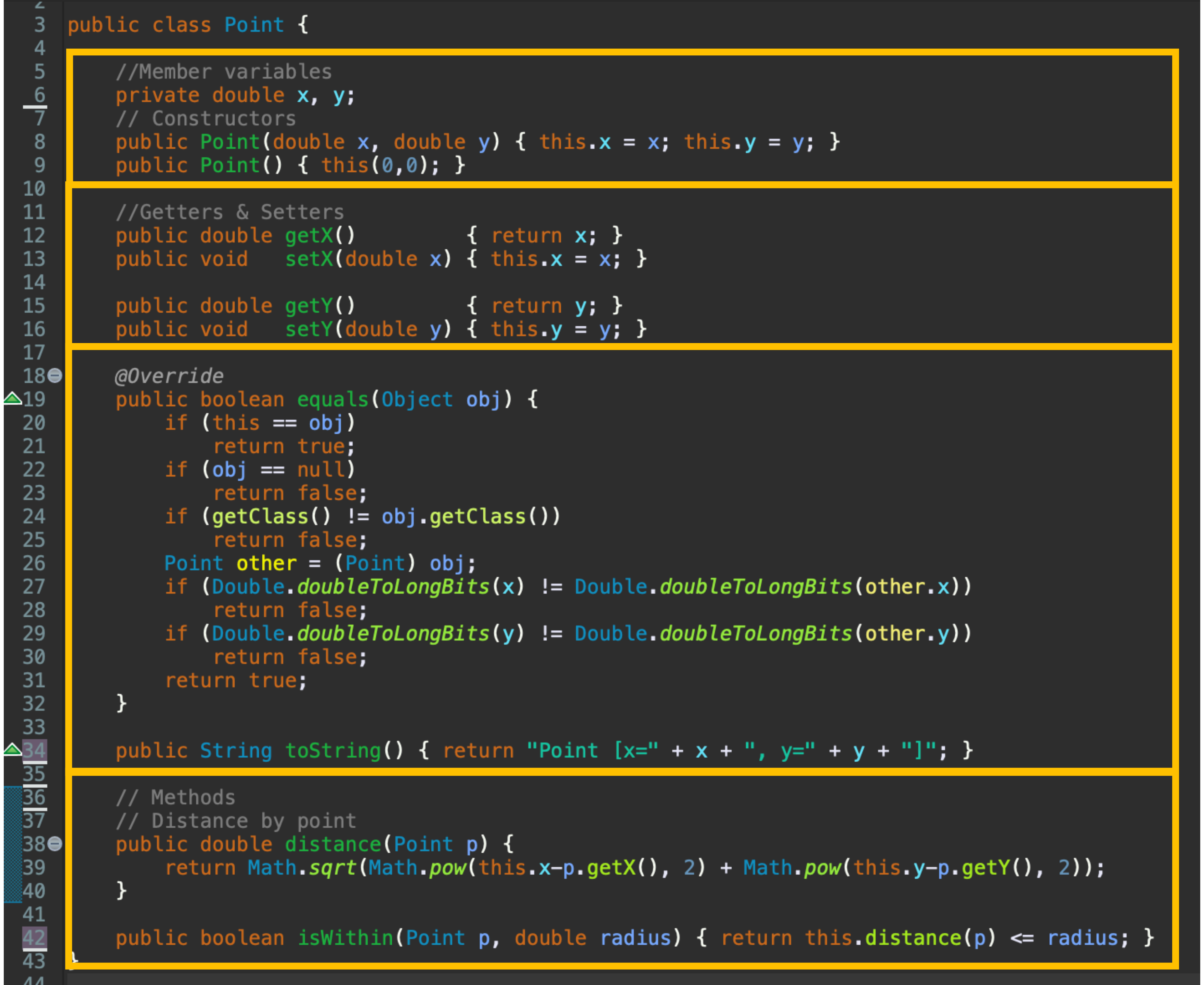
We can also overload constructors in Java, that allows a class to have more than one constructor having different argument lists.



```
// Constructors
public Point(double x, double y) { this.x = x; this.y = y; }
public Point() { this(0,0); }
```

Constructors always need to populate the open member variables.
But we can provide default options.

Points, Polyline, Polygons & Inheritance



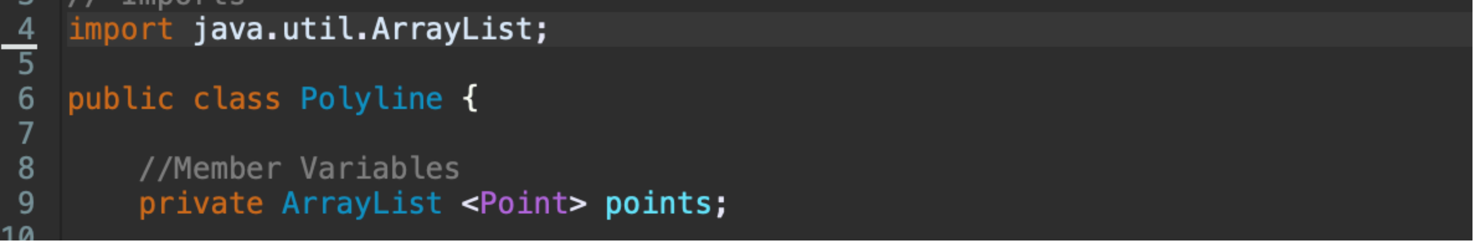
```java
public class Point {

    //Member variables
    private double x, y;
    // Constructors
    public Point(double x, double y) { this.x = x; this.y = y; }
    public Point() { this(0,0); }

    //Getters & Setters
    public double getX()          { return x; }
    public void   setX(double x) { this.x = x; }

    public double getY()          { return y; }
    public void   setY(double y) { this.y = y; }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Point other = (Point) obj;
        if (Double.doubleToLongBits(x) != Double.doubleToLongBits(other.x))
            return false;
        if (Double.doubleToLongBits(y) != Double.doubleToLongBits(other.y))
            return false;
        return true;
    }

    public String toString() { return "Point [x=" + x + ", y=" + y + "]"; }

    // Methods
    // Distance by point
    public double distance(Point p) {
        return Math.sqrt(Math.pow(this.x-p.getX(), 2) + Math.pow(this.y-p.getY(), 2));
    }

    public boolean isWithin(Point p, double radius) { return this.distance(p) <= radius; }
}
```

Member Variables
w/ Overloaded constructor

Getters and setters

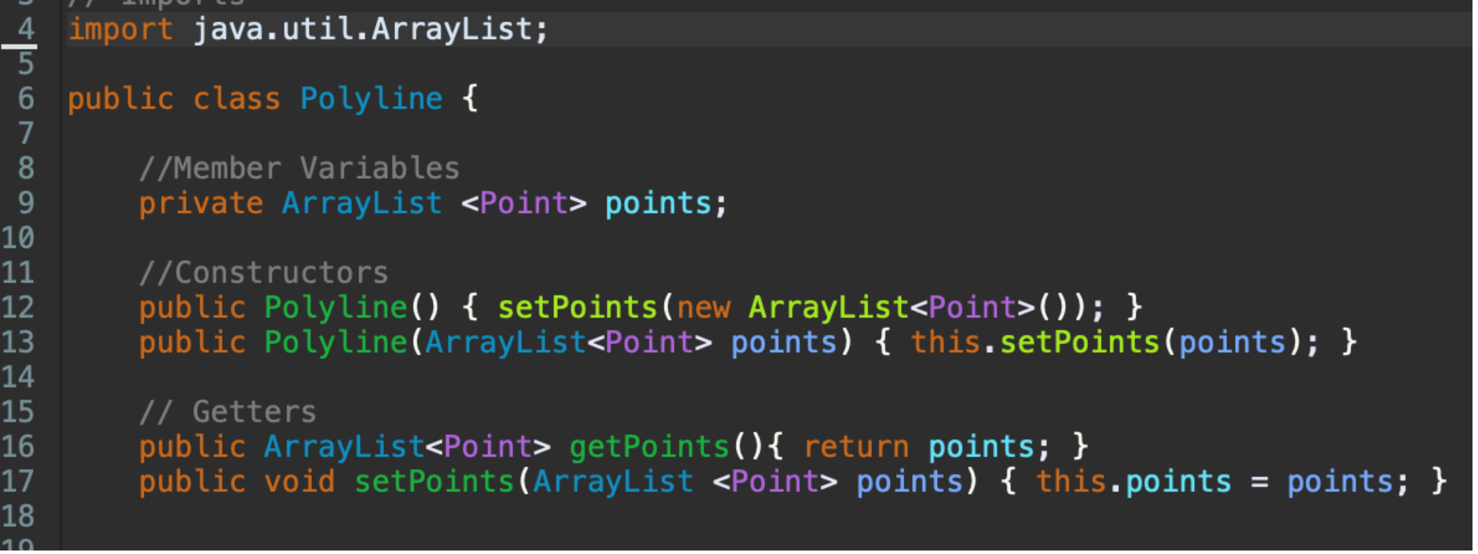
@Overriders

Point Methods



```java
// imports
import java.util.ArrayList;

public class Polyline {

    //Member Variables
    private ArrayList <Point> points;
```

Class named Polyline w/
an open member variable

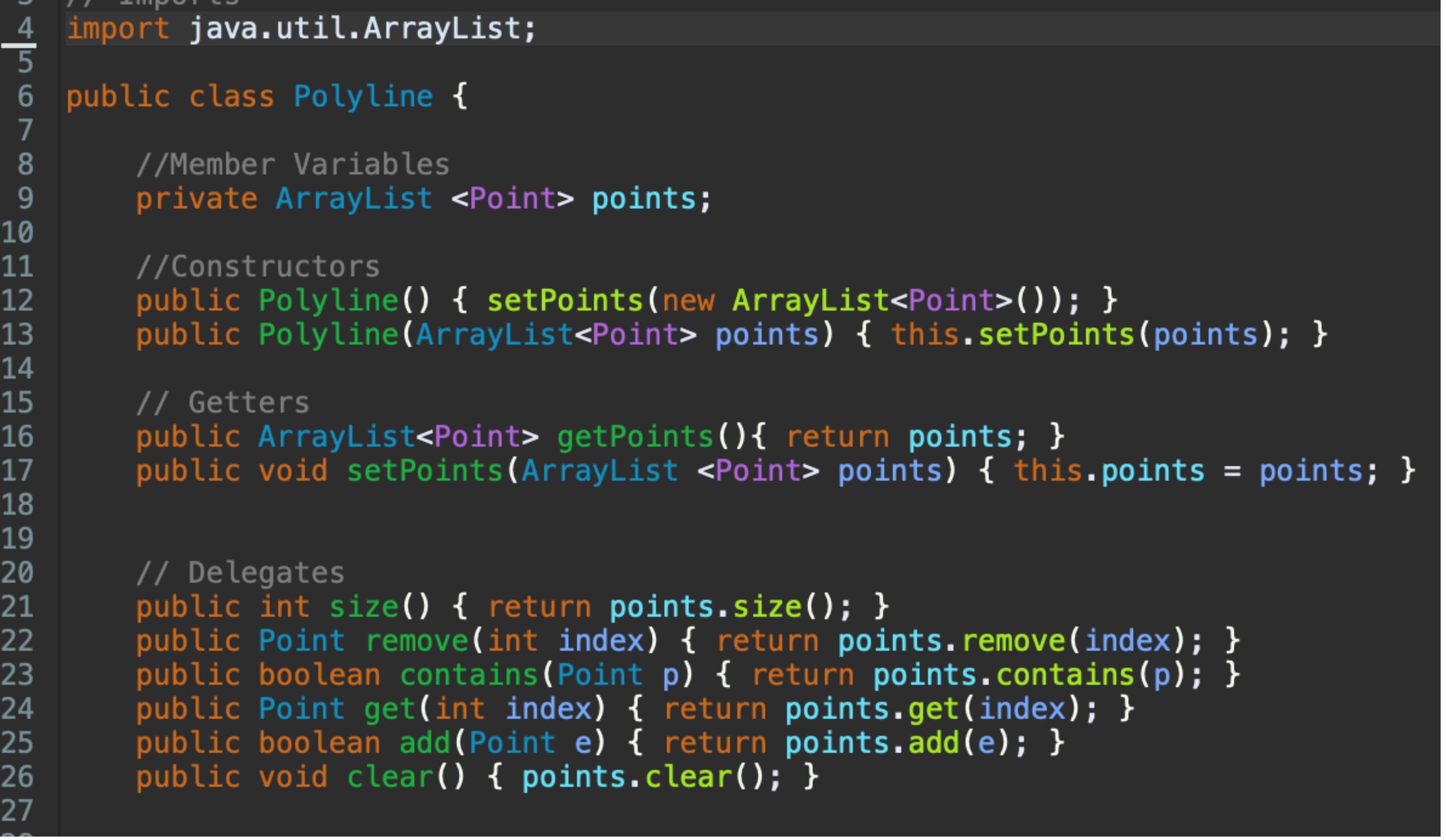


```java
import java.util.ArrayList;

public class Polyline {

    //Member Variables
    private ArrayList <Point> points;

    //Constructors
    public Polyline() { setPoints(new ArrayList<Point>()); }
    public Polyline(ArrayList<Point> points) { this.setPoints(points); }

    // Getters
    public ArrayList<Point> getPoints(){ return points; }
    public void setPoints(ArrayList <Point> points) { this.points = points; }
```

Class named Polyline w/
an open member variable

Create an overloaded constructor that
(A) Takes a set of points –or-
(B) Initializes an empty ArrayList of points

Autogenerate getters and setters



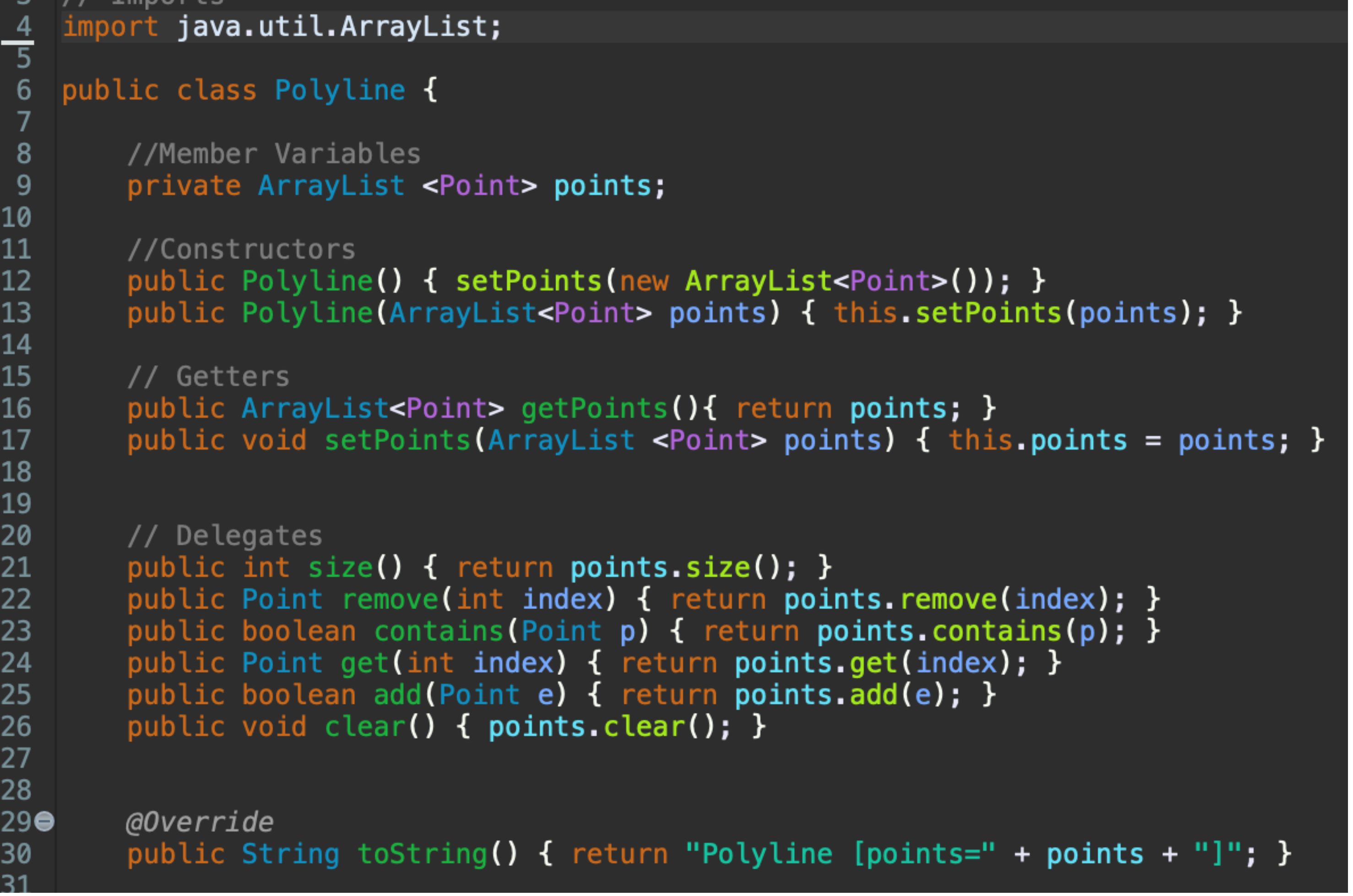
```java
import java.util.ArrayList;

public class Polyline {

    //Member Variables
    private ArrayList <Point> points;

    //Constructors
    public Polyline() { setPoints(new ArrayList<Point>()); }
    public Polyline(ArrayList<Point> points) { this.setPoints(points); }

    // Getters
    public ArrayList<Point> getPoints(){ return points; }
    public void setPoints(ArrayList <Point> points) { this.points = points; }


    // Delegates
    public int size() { return points.size(); }
    public Point remove(int index) { return points.remove(index); }
    public boolean contains(Point p) { return points.contains(p); }
    public Point get(int index) { return points.get(index); }
    public boolean add(Point e) { return points.add(e); }
    public void clear() { points.clear(); }

```

Class named Polyline w/
an open member variable

Create an overloaded constructor that
(A) Takes a set of points –or-
(B) Initializes an empty ArrayList of points

Autogenerate getters and setters

Auto generate delegators for the
Point ArrayList



```java
import java.util.ArrayList;

public class Polyline {

    //Member Variables
    private ArrayList <Point> points;

    //Constructors
    public Polyline() { setPoints(new ArrayList<Point>()); }
    public Polyline(ArrayList<Point> points) { this.setPoints(points); }

    // Getters
    public ArrayList<Point> getPoints(){ return points; }
    public void setPoints(ArrayList <Point> points) { this.points = points; }


    // Delegates
    public int size() { return points.size(); }
    public Point remove(int index) { return points.remove(index); }
    public boolean contains(Point p) { return points.contains(p); }
    public Point get(int index) { return points.get(index); }
    public boolean add(Point e) { return points.add(e); }
    public void clear() { points.clear(); }


    @Override
    public String toString() { return "Polyline [points=" + points + "]"; }
```

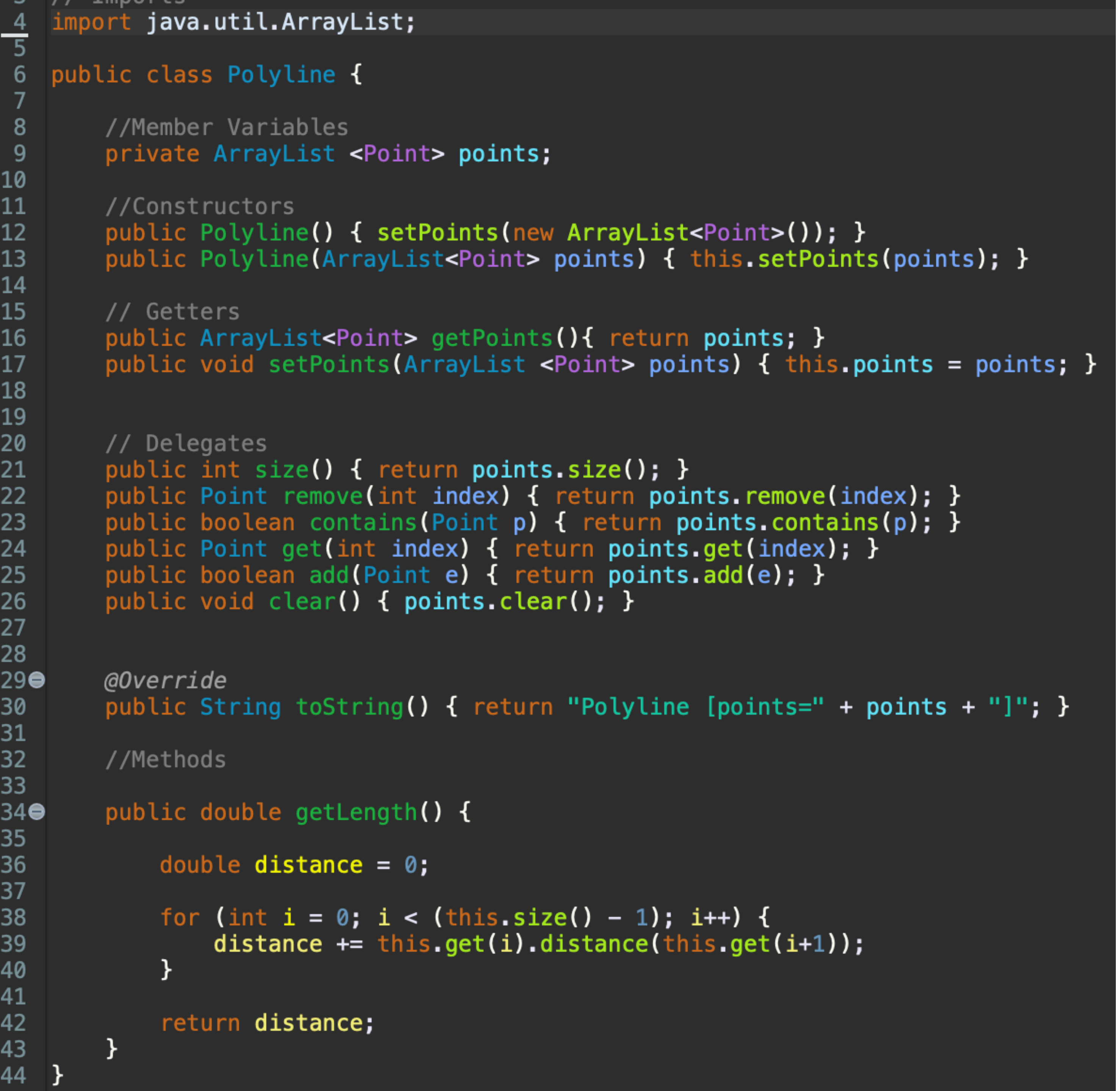
Class named Polyline w/
an open member variable

Create an overloaded constructor that
(A) Takes a set of points –or-
(B) Initializes an empty ArrayList of points

Autogenerate getters and setters

Auto generate delegators for the
Point ArrayList

Override the toString() method



```java
import java.util.ArrayList;

public class Polyline {

    //Member Variables
    private ArrayList <Point> points;

    //Constructors
    public Polyline() { setPoints(new ArrayList<Point>()); }
    public Polyline(ArrayList<Point> points) { this.setPoints(points); }

    // Getters
    public ArrayList<Point> getPoints(){ return points; }
    public void setPoints(ArrayList <Point> points) { this.points = points; }


    // Delegates
    public int size() { return points.size(); }
    public Point remove(int index) { return points.remove(index); }
    public boolean contains(Point p) { return points.contains(p); }
    public Point get(int index) { return points.get(index); }
    public boolean add(Point e) { return points.add(e); }
    public void clear() { points.clear(); }


    @Override
    public String toString() { return "Polyline [points=" + points + "]"; }

    //Methods

    public double getLength() {

        double distance = 0;

        for (int i = 0; i < (this.size() - 1); i++) {
            distance += this.get(i).distance(this.get(i+1));
        }

        return distance;
    }
}
```

Class named Polyline w/
an open member variable

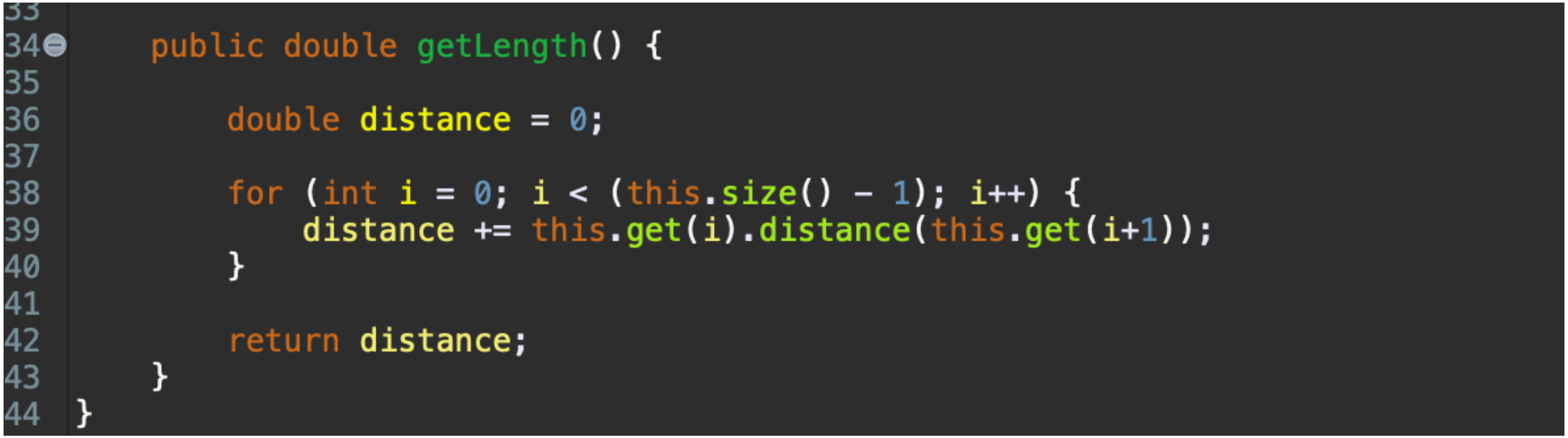
Create an overloaded constructor that
(A) Takes a set of points –or-
(B) Initializes an empty ArrayList of points

Autogenerate getters and setters

Auto generate delegators for the
Point ArrayList

Override the toString() method

Lets right a new method and revisit the for-loop



```
33
34      public double getLength() {
35
36          double distance = 0;
37
38          for (int i = 0; i < (this.size() - 1); i++) {
39              distance += this.get(i).distance(this.get(i+1));
40          }
41
42          return distance;
43      }
44  }
```

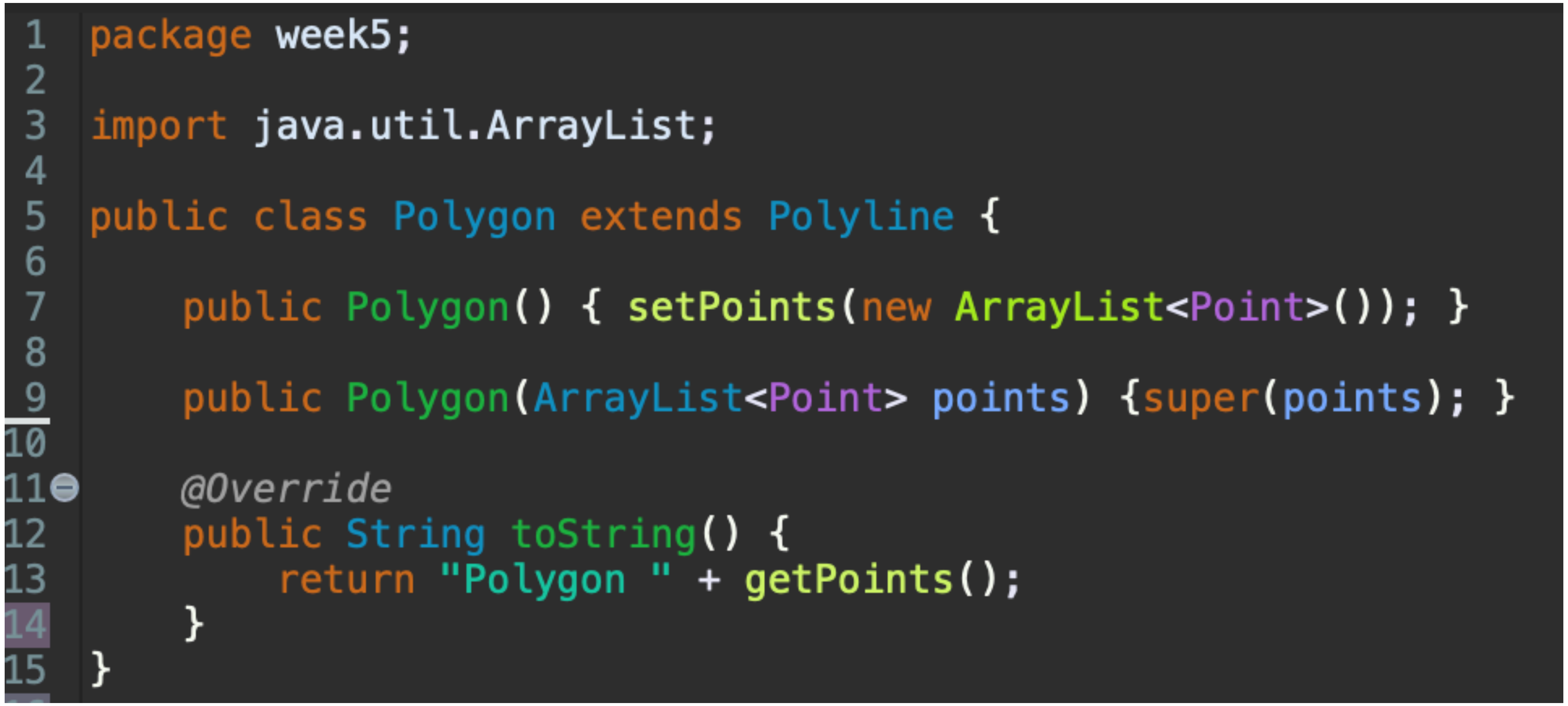
Line 34: This is a public method named getLength() that returns a double and requires no input
Line 36: Initialize a double variable called distance and set its initial value to 0
Line 38: Initialize a for loop that goes from 0 to the size of the Polyline – 1
Line 39: Take the current distance value and add the distance between point i and point i+1
Line 42: When the loop finishes return distance!

Polygon vs PolyLine?

What do we know about Polygons and their relation to Polylines?

- Polygons are Polylines that have an equal start and end point
- Both are simply collections of Points.
- Everything we can do with a Polyline, we can do with a Polygon.

This is a perfect opportunity to define a Polygon class that inherits the characteristics of a Polyline!

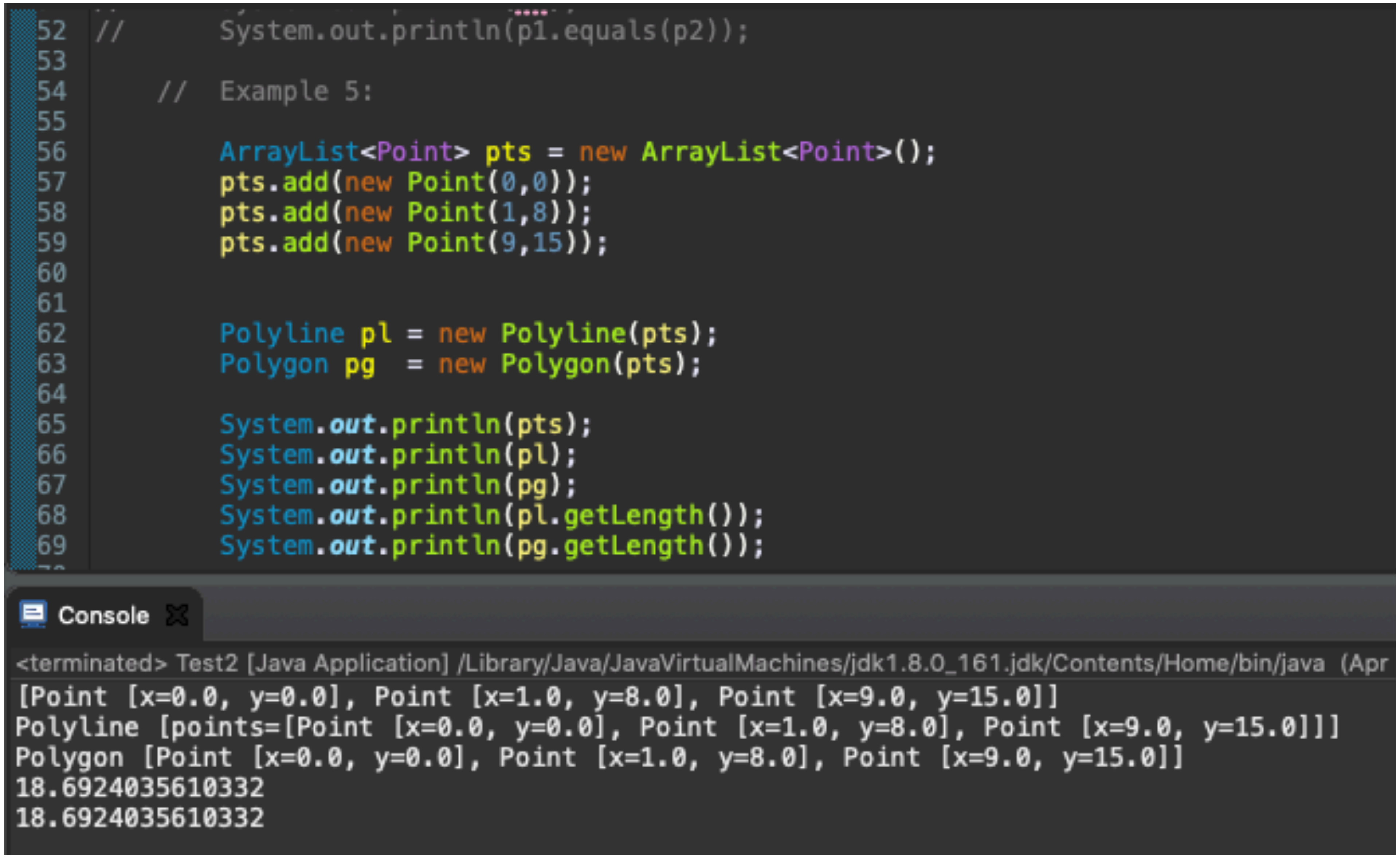


```
1  package week5;
2
3  import java.util.ArrayList;
4
5  public class Polygon extends Polyline {
6
7      public Polygon() { setPoints(new ArrayList<Point>()); }
8
9      public Polygon(ArrayList<Point> points) {super(points); }
10
11     @Override
12     public String toString() {
13         return "Polygon " + getPoints();
14     }
15 }
```

Line 5: Polygon is a class that extends the Polyline Class.
This means that Polygon Inherits all aspects from Polyline and that all Polyline Methods are accessible to Polygon objects

Line 7: We still need constructors and here we overload the constructor allowing users to create empty Polygon Objects.
Line 9: Or provide an ArrayList of points that fills the open points variable of the parent (super) class Polyline

Line 12: Now we don’t want Polygons objects to inherit the toString() over rider of Polyline so we over-ride the over-ride

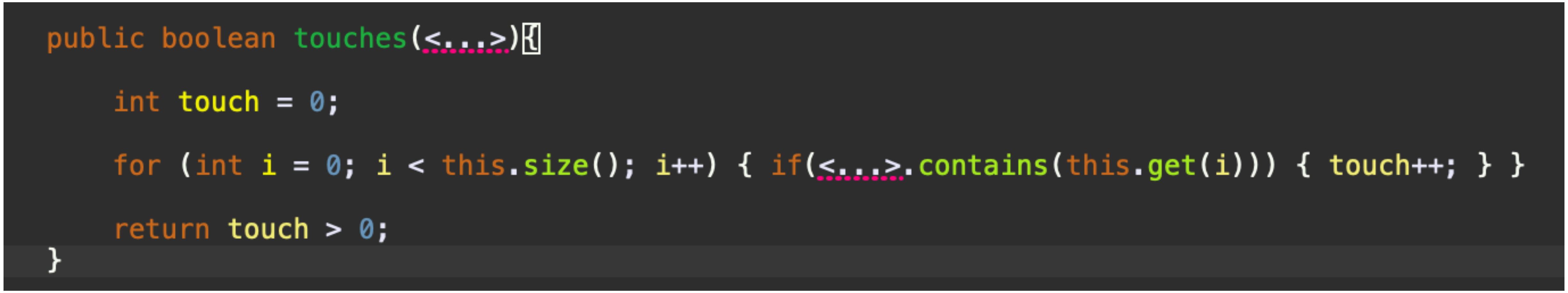


```
//        System.out.println(p1.equals(p2));

     //   Example 5:

          ArrayList<Point> pts = new ArrayList<Point>();
          pts.add(new Point(0,0));
          pts.add(new Point(1,8));
          pts.add(new Point(9,15));


          Polyline pl = new Polyline(pts);
          Polygon pg  = new Polygon(pts);

          System.out.println(pts);
          System.out.println(pl);
          System.out.println(pg);
          System.out.println(pl.getLength());
          System.out.println(pg.getLength());
```

Console

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr
[Point [x=0.0, y=0.0], Point [x=1.0, y=8.0], Point [x=9.0, y=15.0]]
Polyline [points=[Point [x=0.0, y=0.0], Point [x=1.0, y=8.0], Point [x=9.0, y=15.0]]]
Polygon [Point [x=0.0, y=0.0], Point [x=1.0, y=8.0], Point [x=9.0, y=15.0]]
18.6924035610332
18.6924035610332
```

Let’s make a new method that looks if two geometries are touching:



```
public boolean touches(<...>){
    int touch = 0;
    for (int i = 0; i < this.size(); i++) { if(<...>.contains(this.get(i))) { touch++; } }
    return touch > 0;
}
```

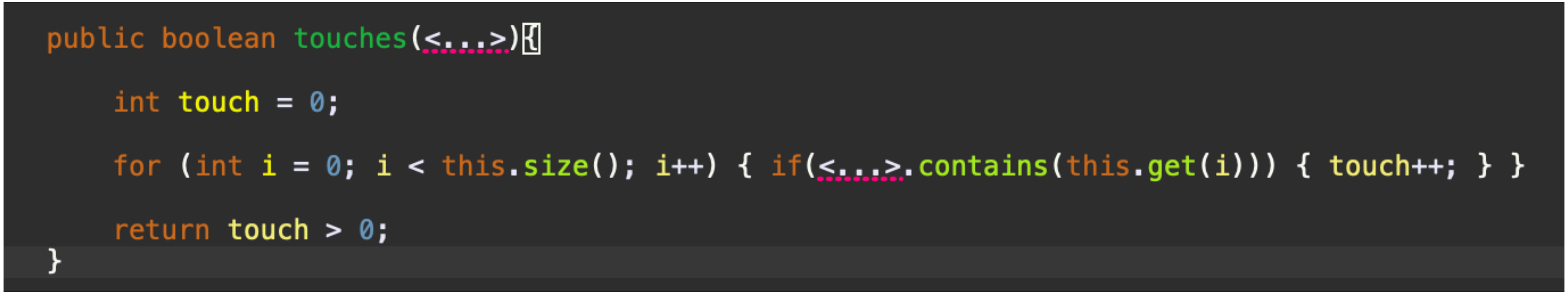
Here we make a public class called touches that take some input <…>

It initializes an integer called touch with a starting value of 0

It opens a for loop that runs along the size of the object the method is applied to (this).
It opens a conditional if statement that says:

> Does the input <…> contain the first element of the the object this method is applied too (this)
> If TRUE then increase ‘’touch” by 1, otherwise, do nothing

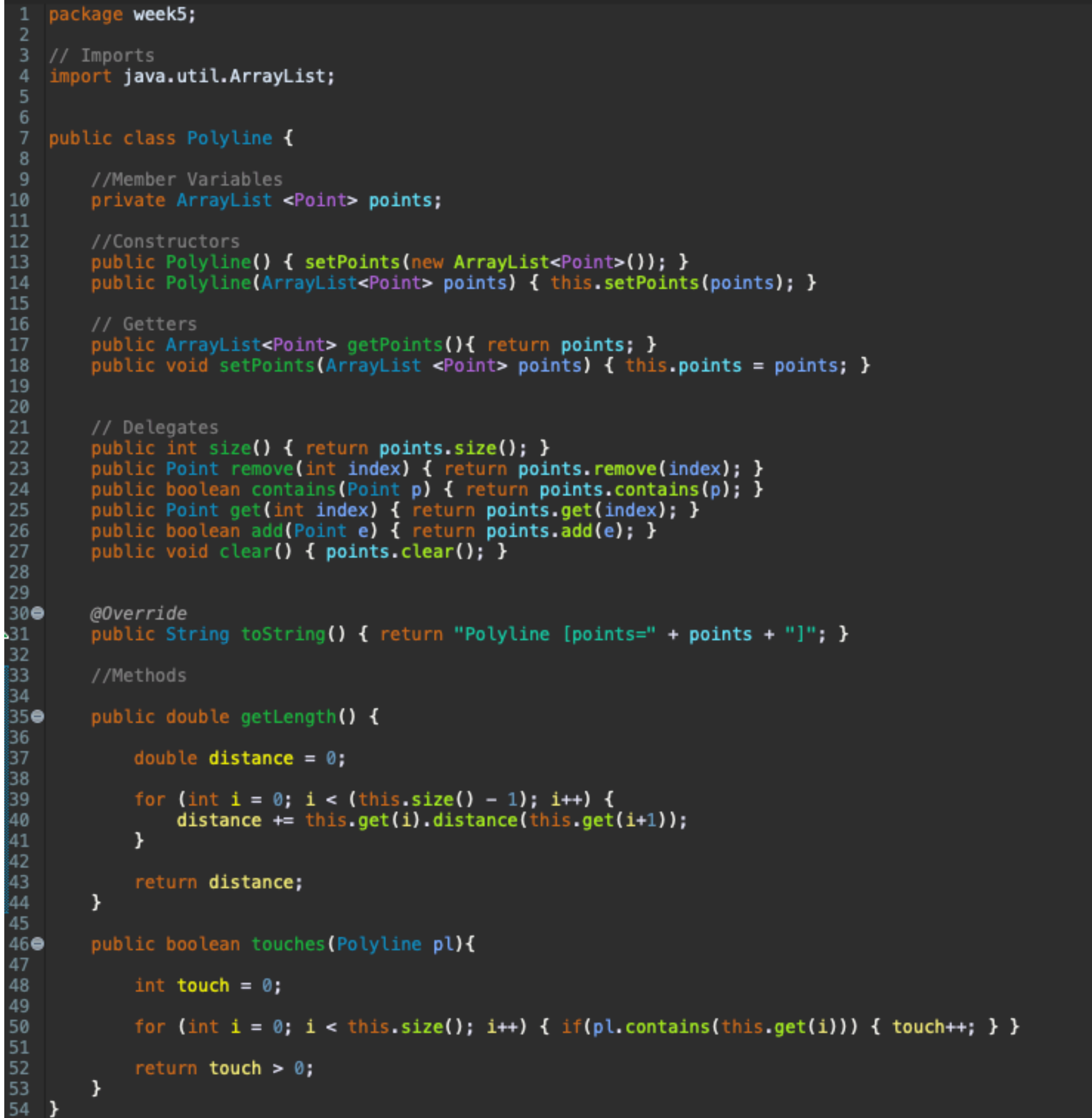
Once its all done, check if touch is greater then 0 and if so return TRUE, else return FALSE



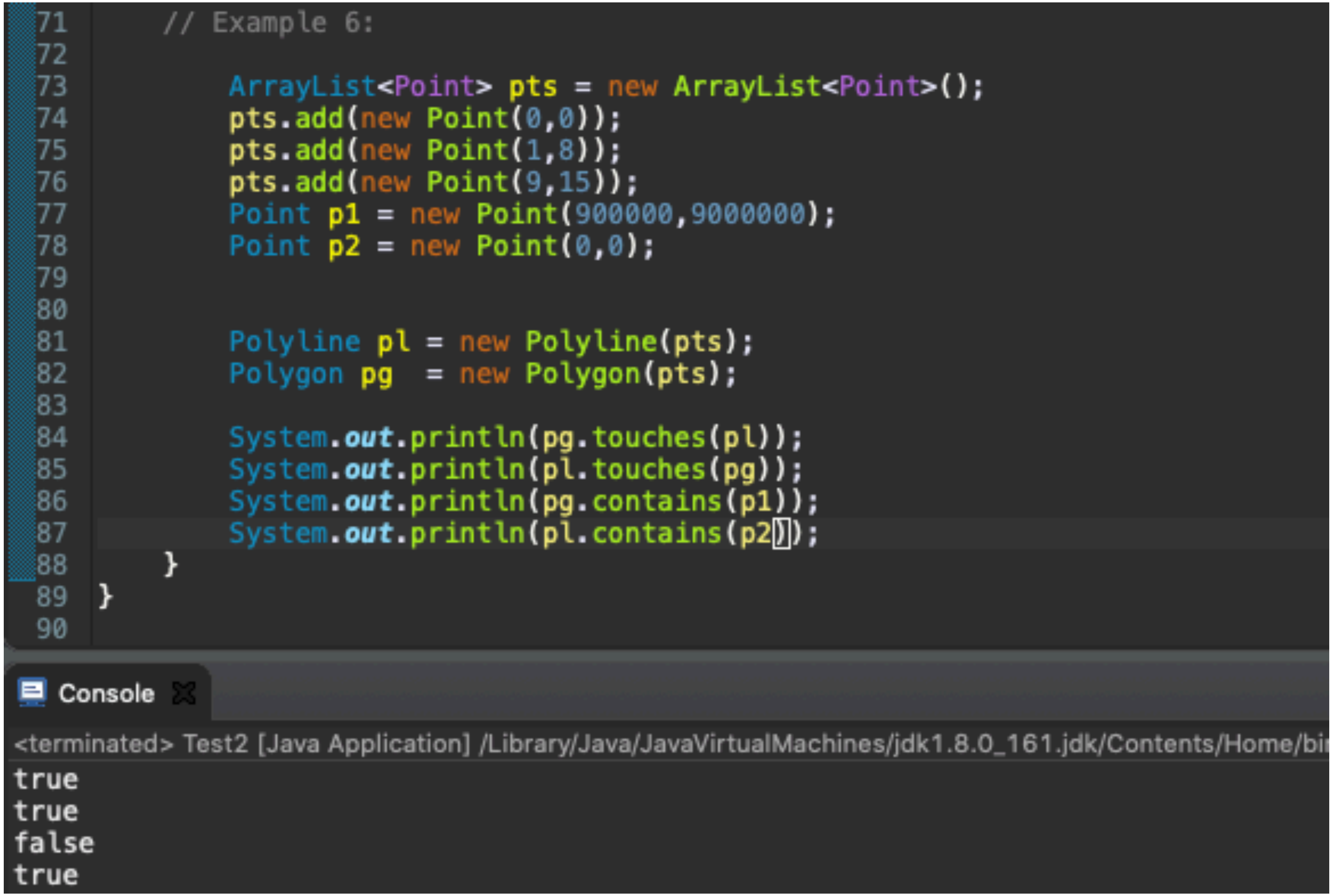
```
public boolean touches(<...>){

    int touch = 0;

    for (int i = 0; i < this.size(); i++) { if(<...>.contains(this.get(i))) { touch++; } }

    return touch > 0;
}
```

1. What class of object should we pass as input ?
2. Where should this method go?

Put touches in Polyline



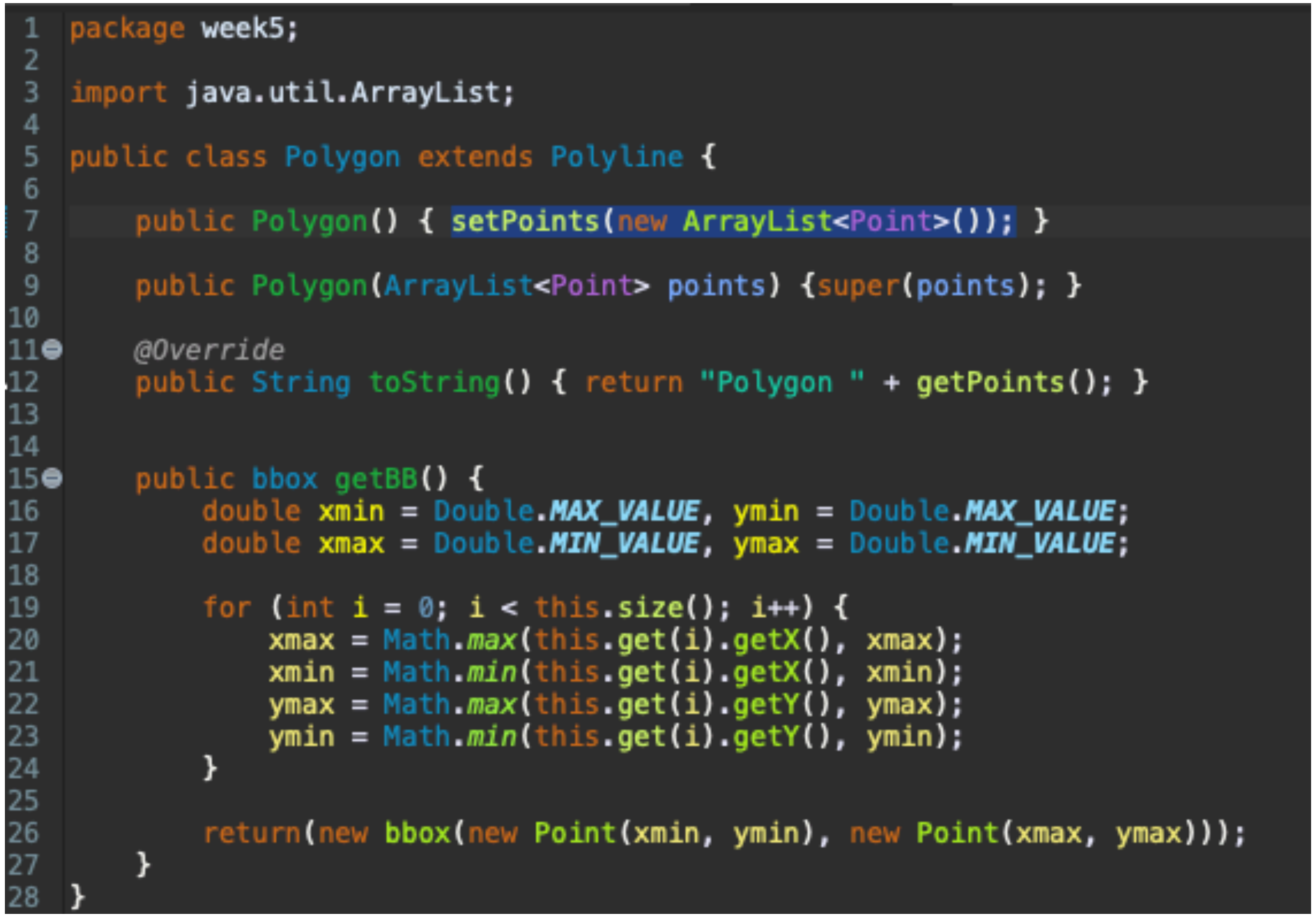
```java
package week5;

// Imports
import java.util.ArrayList;


public class Polyline {

    //Member Variables
    private ArrayList <Point> points;

    //Constructors
    public Polyline() { setPoints(new ArrayList<Point>()); }
    public Polyline(ArrayList<Point> points) { this.setPoints(points); }

    // Getters
    public ArrayList<Point> getPoints(){ return points; }
    public void setPoints(ArrayList <Point> points) { this.points = points; }


    // Delegates
    public int size() { return points.size(); }
    public Point remove(int index) { return points.remove(index); }
    public boolean contains(Point p) { return points.contains(p); }
    public Point get(int index) { return points.get(index); }
    public boolean add(Point e) { return points.add(e); }
    public void clear() { points.clear(); }


    @Override
    public String toString() { return "Polyline [points=" + points + "]"; }

    //Methods

    public double getLength() {

        double distance = 0;

        for (int i = 0; i < (this.size() - 1); i++) {
            distance += this.get(i).distance(this.get(i+1));
        }

        return distance;
    }

    public boolean touches(Polyline pl){

        int touch = 0;

        for (int i = 0; i < this.size(); i++) { if(pl.contains(this.get(i))) { touch++; } }

        return touch > 0;
    }
}
```



```
    // Example 6:

        ArrayList<Point> pts = new ArrayList<Point>();
        pts.add(new Point(0,0));
        pts.add(new Point(1,8));
        pts.add(new Point(9,15));
        Point p1 = new Point(900000,9000000);
        Point p2 = new Point(0,0);


        Polyline pl = new Polyline(pts);
        Polygon pg  = new Polygon(pts);

        System.out.println(pg.touches(pl));
        System.out.println(pl.touches(pg));
        System.out.println(pg.contains(p1));
        System.out.println(pl.contains(p2));
    }
}

```

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bi
true
true
false
true
```

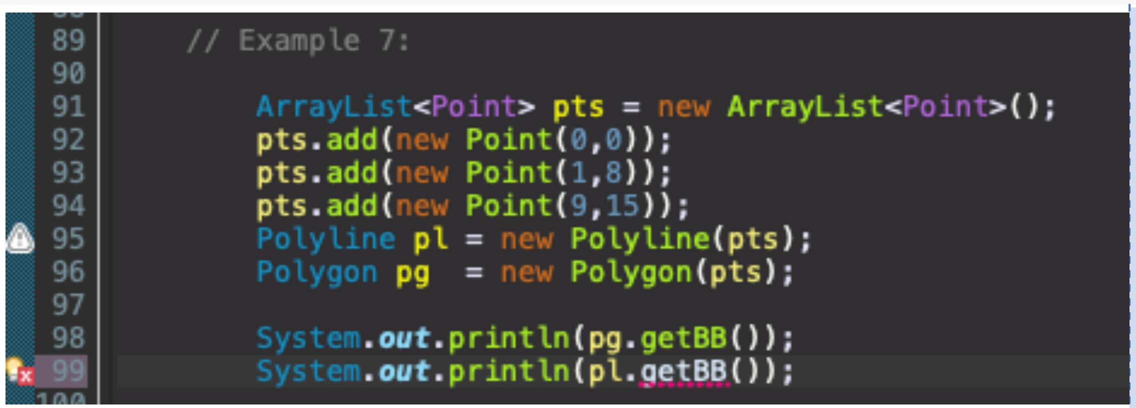
Lets make an method in Polygon called getBB



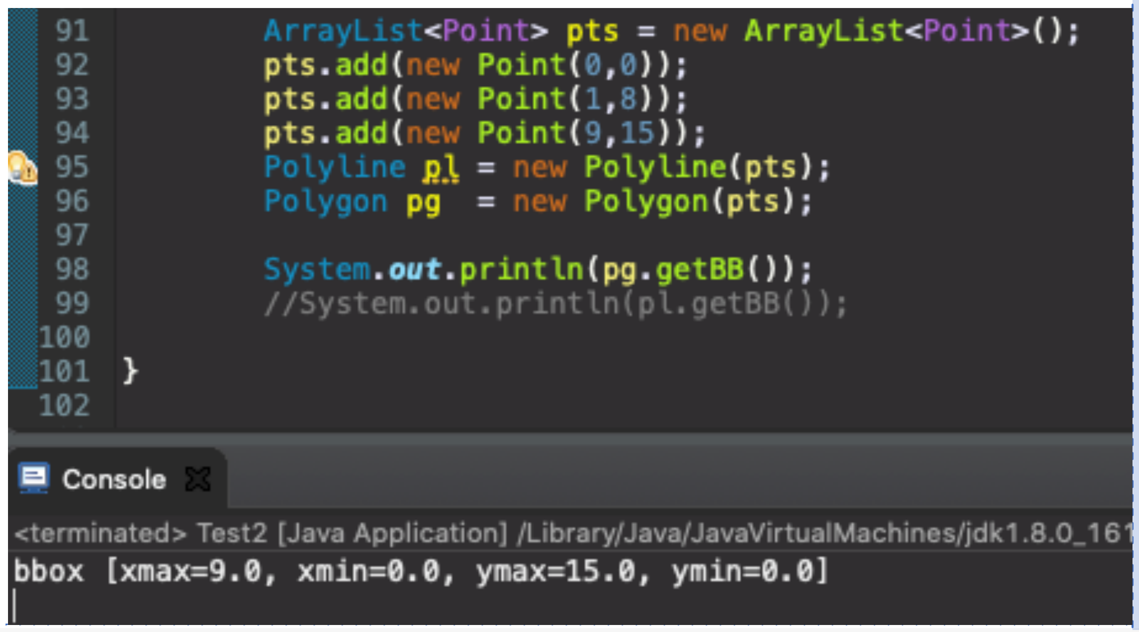
```java
package week5;

import java.util.ArrayList;

public class Polygon extends Polyline {

    public Polygon() { setPoints(new ArrayList<Point>()); }

    public Polygon(ArrayList<Point> points) {super(points); }

    @Override
    public String toString() { return "Polygon " + getPoints(); }


    public bbox getBB() {
        double xmin = Double.MAX_VALUE, ymin = Double.MAX_VALUE;
        double xmax = Double.MIN_VALUE, ymax = Double.MIN_VALUE;

        for (int i = 0; i < this.size(); i++) {
            xmax = Math.max(this.get(i).getX(), xmax);
            xmin = Math.min(this.get(i).getX(), xmin);
            ymax = Math.max(this.get(i).getY(), ymax);
            ymin = Math.min(this.get(i).getY(), ymin);
        }

        return(new bbox(new Point(xmin, ymin), new Point(xmax, ymax)));
    }
}
```

*There are certainty arguments that this should go in Polyline, and it probably should.
But for example, we are going to keep it in Polygon…

Example 7



```
    // Example 7:

        ArrayList<Point> pts = new ArrayList<Point>();
        pts.add(new Point(0,0));
        pts.add(new Point(1,8));
        pts.add(new Point(9,15));
        Polyline pl = new Polyline(pts);
        Polygon pg  = new Polygon(pts);

        System.out.println(pg.getBB());
        System.out.println(pl.getBB());
```

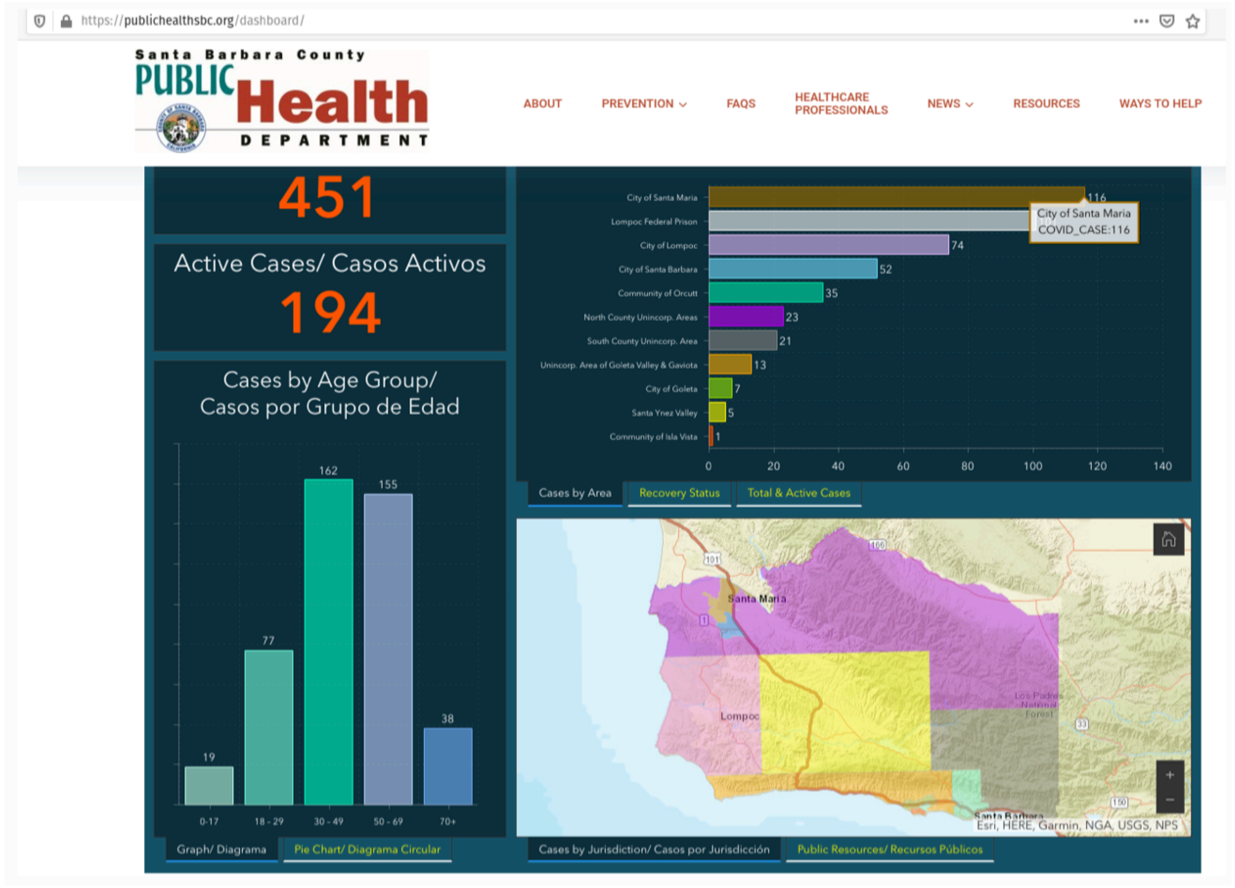


```
        ArrayList<Point> pts = new ArrayList<Point>();
        pts.add(new Point(0,0));
        pts.add(new Point(1,8));
        pts.add(new Point(9,15));
        Polyline pl = new Polyline(pts);
        Polygon pg  = new Polygon(pts);

        System.out.println(pg.getBB());
        //System.out.println(pl.getBB());

}

```

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161
bbox [xmax=9.0, xmin=0.0, ymax=15.0, ymin=0.0]
```

Our COVID simulations

- How does this all relate??


https://publichealthsbc.org/dashboard/
Santa Barbara County PUBLIC Health DEPARTMENT
ABOUT
PREVENTION
FAQS
HEALTHCARE PROFESSIONALS
NEWS
RESOURCES
WAYS TO HELP
451
Active Cases/ Casos Activos
194
Cases by Age Group/ Casos por Grupo de Edad
0-17: 19
18 - 29: 77
30 - 49: 162
50 - 69: 155
70+: 38
City of Santa Maria: 116
Lompoc Federal Prison
City of Lompoc: 74
City of Santa Barbara: 52
Community of Orcutt: 35
North County Unincorp. Areas: 23
South County Unincorp. Area: 21
Unincorp. Area of Goleta Valley & Gaviota: 13
City of Goleta: 7
Santa Ynez Valley: 5
Community of Isla Vista: 1
City of Santa Maria COVID_CASE:116
Cases by Area
Recovery Status
Total & Active Cases
Santa Maria
Lompoc
Santa Barbara
Esri, HERE, Garmin, NGA, USGS, NPS
Graph/ Diagrama
Pie Chart/ Diagrama Circular
Cases by Jurisdiction/ Casos por Jurisdicción
Public Resources/ Recursos Públicos

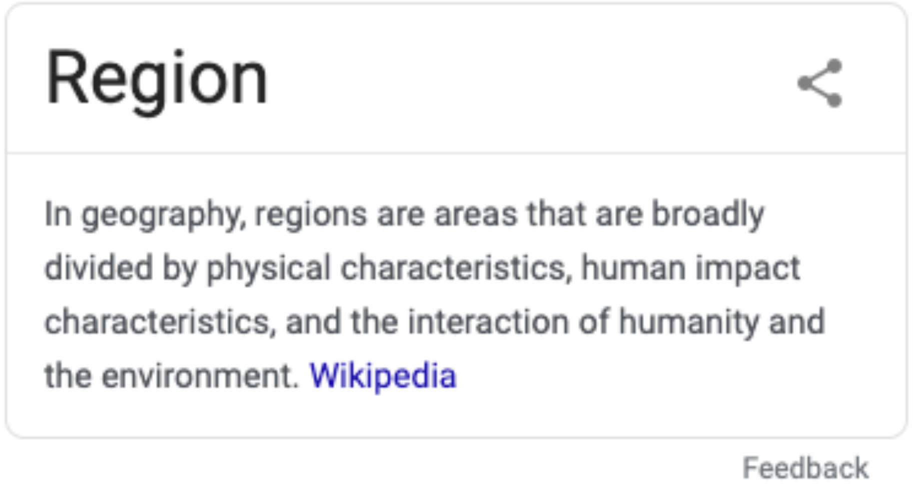
What have we done?

- We made a POINT class
- We defined a Polyline class as a collection of points with explicit methods
- We wrote methods that check if Polylines touch
- We extended Polyline to create Polygon
- We added a Polygon method to coherce a Polygon into a bbox object from Polygon(Polyline) Point ArrayList

Regions

All Regions have:

1. a name
2. COVID count
3. Footprint
4. County
5. People (optional)

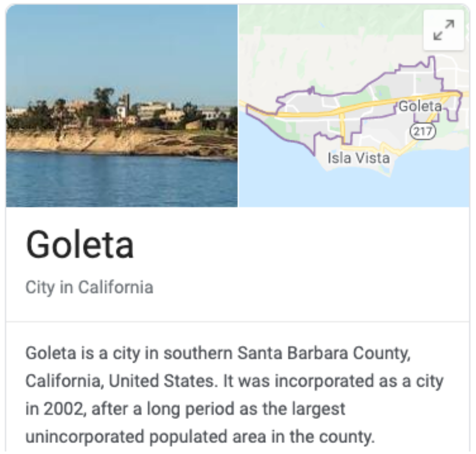


Region

In geography, regions are areas that are broadly divided by physical characteristics, human impact characteristics, and the interaction of humanity and the environment. Wikipedia

Feedback

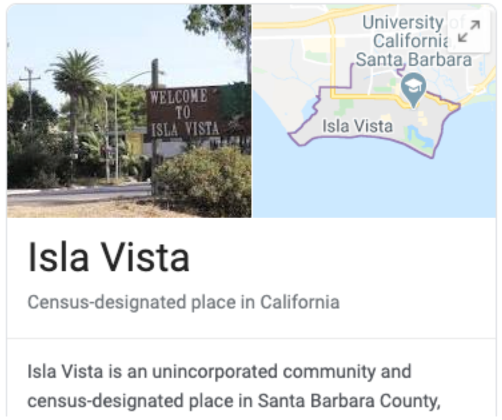
Some regions are **cities**, *some* are **unincorporated** areas:



Goleta

City in California

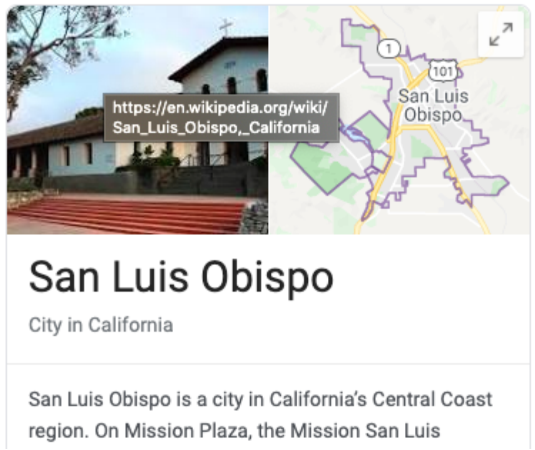
Goleta is a city in southern Santa Barbara County, California, United States. It was incorporated as a city in 2002, after a long period as the largest unincorporated populated area in the county.



Isla Vista

Census-designated place in California

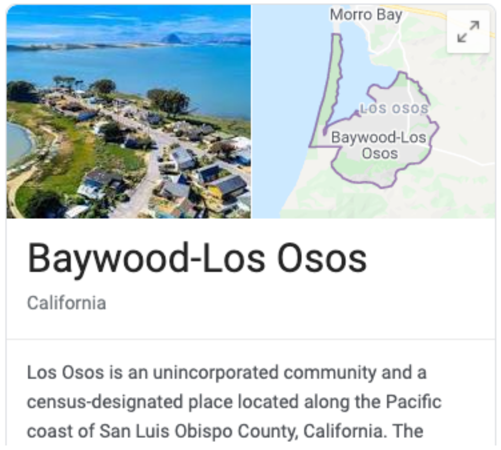
Isla Vista is an unincorporated community and census-designated place in Santa Barbara County,



San Luis Obispo

City in California

San Luis Obispo is a city in California's Central Coast region. On Mission Plaza, the Mission San Luis

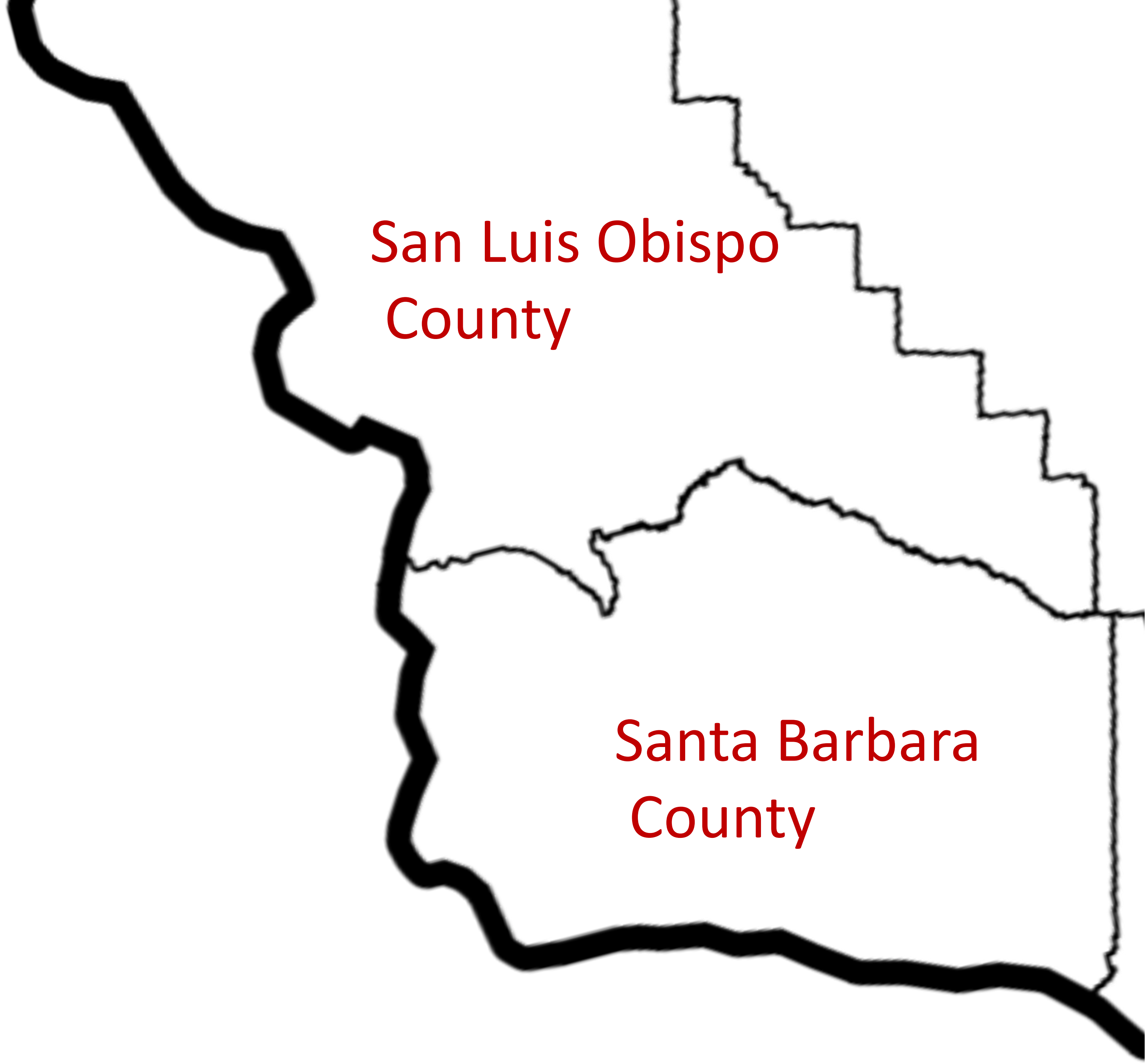


Baywood-Los Osos

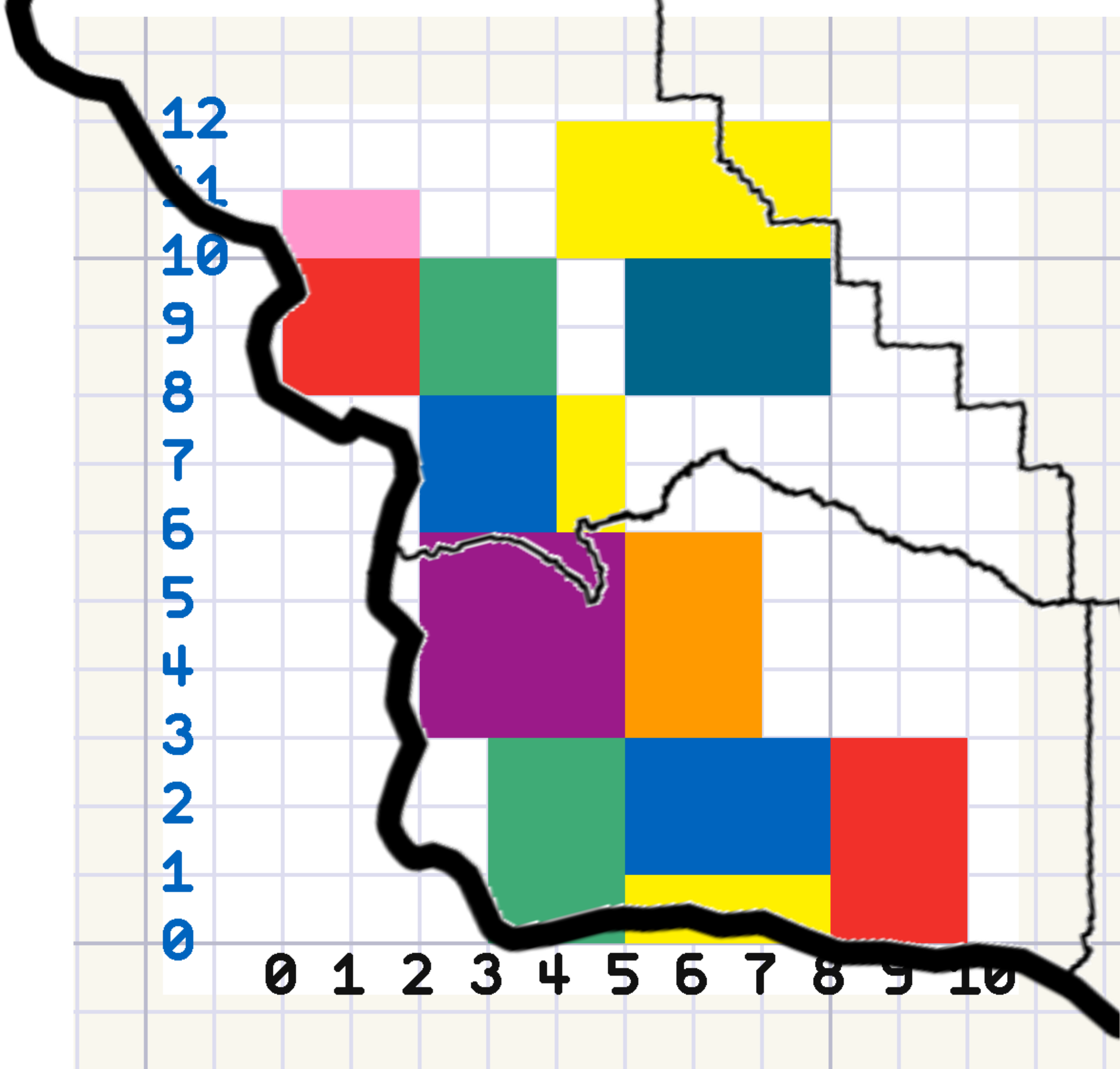
California

Los Osos is an unincorporated community and a census-designated place located along the Pacific coast of San Luis Obispo County, California. The

Imagine our real world

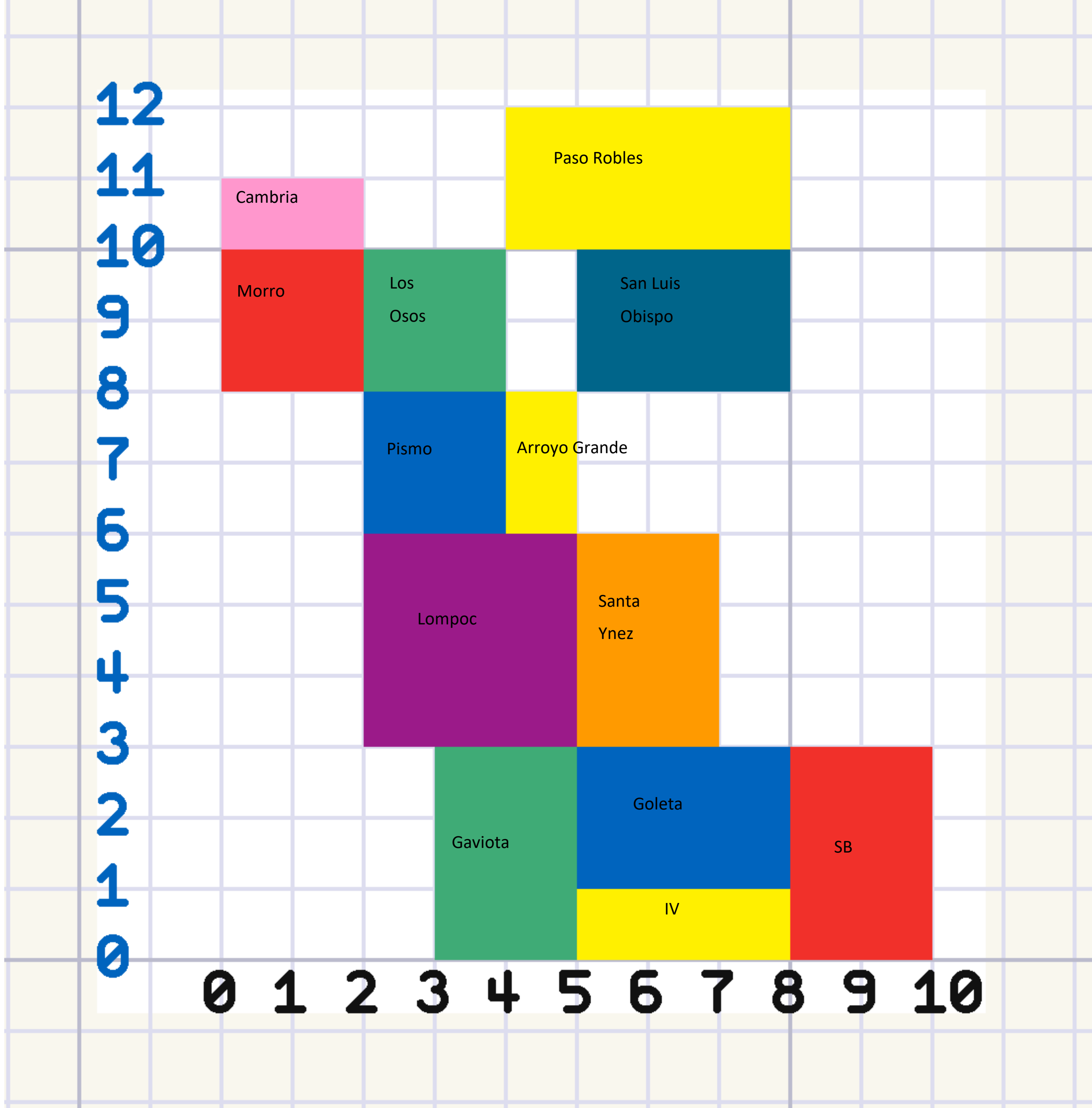


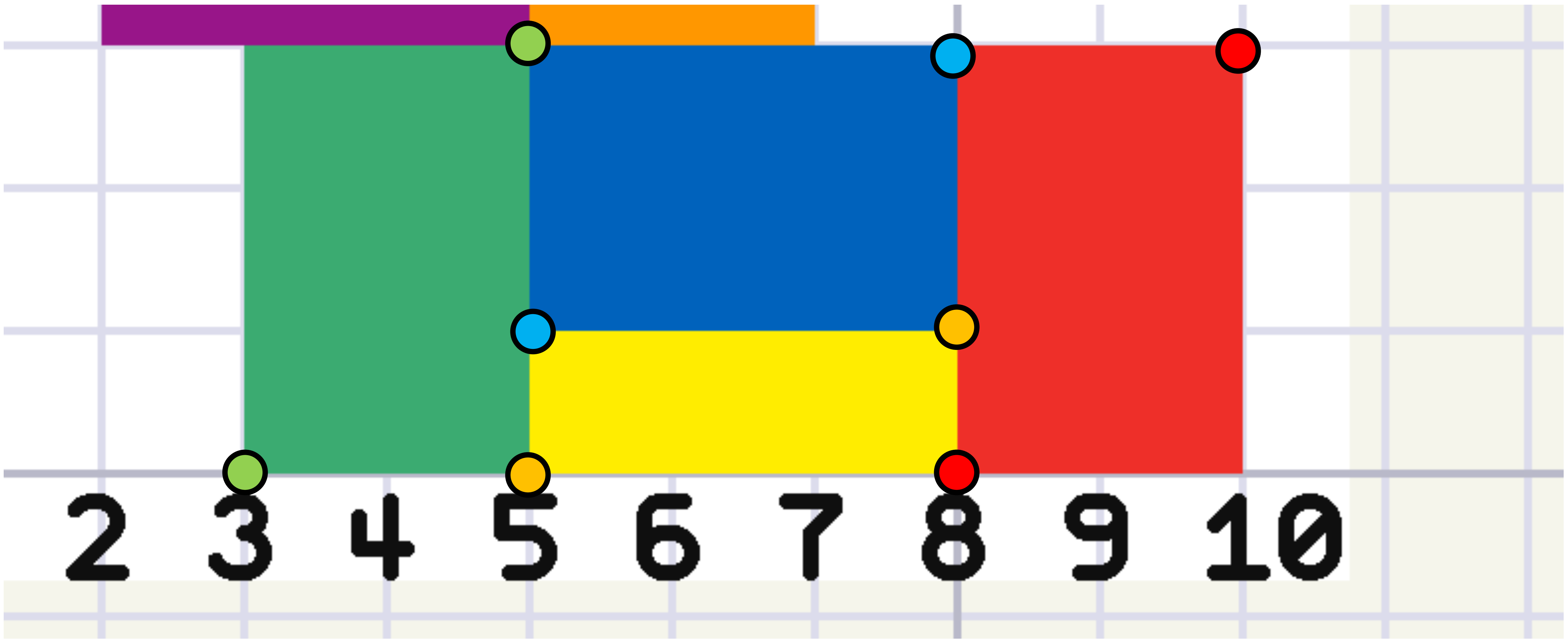
- Each of these counties has a number of regions …



These regions can be imagined as bounding boxes, in cartision space…

BTW does this remind you or Projected Coordinates Systems with false origins and unit increments?



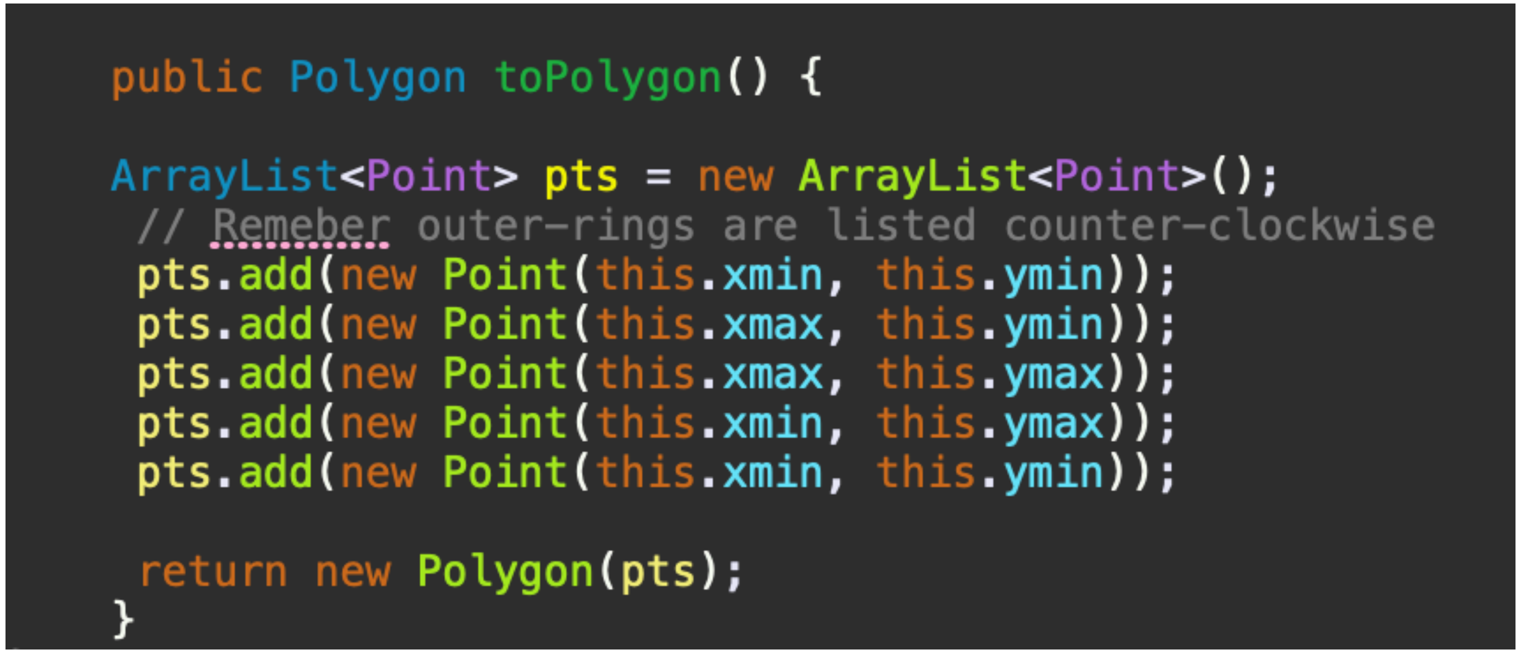


Not a single shared vertex across Bbox representations!!

BBOXs are easier and more efficient to describe, but …

We need full polygon representations to compare vertices ….

Pseudo-Casting…. Put this in the bbox class…



```
public Polygon toPolygon() {

ArrayList<Point> pts = new ArrayList<Point>();
 // Remeber outer-rings are listed counter-clockwise
 pts.add(new Point(this.xmin, this.ymin));
 pts.add(new Point(this.xmax, this.ymin));
 pts.add(new Point(this.xmax, this.ymax));
 pts.add(new Point(this.xmin, this.ymax));
 pts.add(new Point(this.xmin, this.ymin));

 return new Polygon(pts);
}
```

Lets code together!

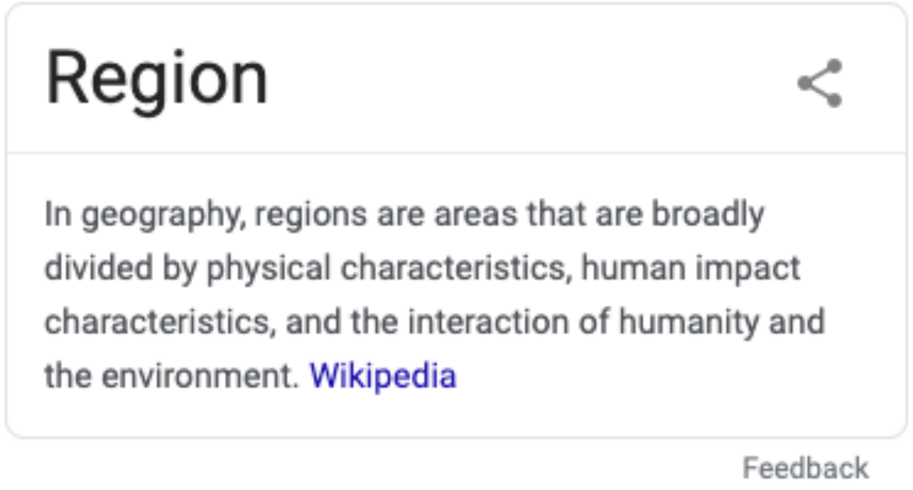
Remember the classes we have discussed are all available on the Github page under week5

Start by making a Region class

Regions

All Regions have:

1. a name
2. COVID count
3. Footprint
4. County
5. People (optional)

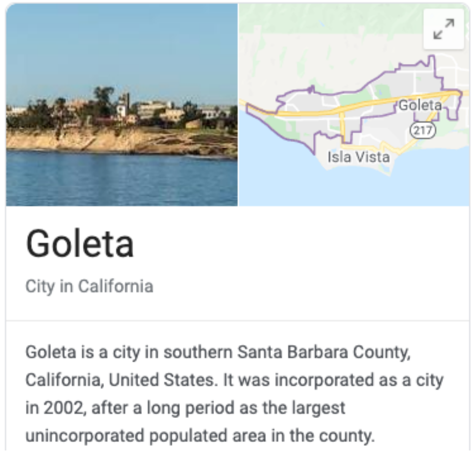


Region

In geography, regions are areas that are broadly divided by physical characteristics, human impact characteristics, and the interaction of humanity and the environment. Wikipedia

Feedback

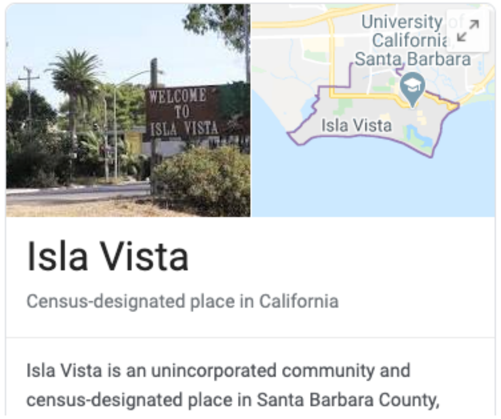
Some regions are **cities**, *some* are **unincorporated** areas:



Goleta

City in California

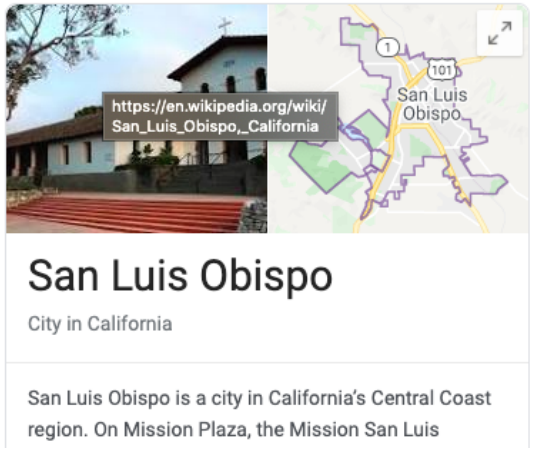
Goleta is a city in southern Santa Barbara County, California, United States. It was incorporated as a city in 2002, after a long period as the largest unincorporated populated area in the county.



Isla Vista

Census-designated place in California

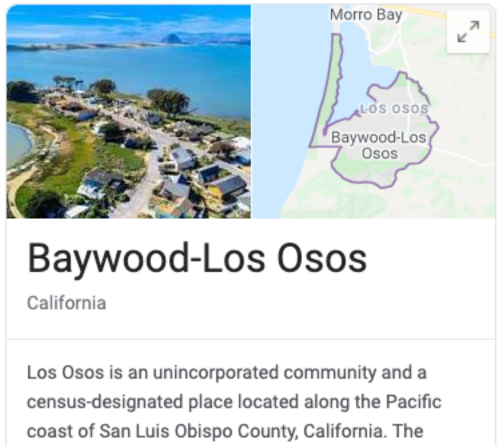
Isla Vista is an unincorporated community and census-designated place in Santa Barbara County,



San Luis Obispo

City in California

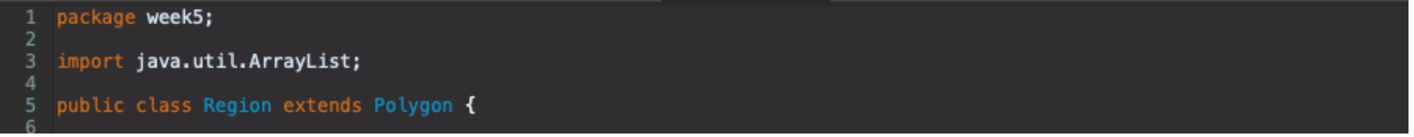
San Luis Obispo is a city in California's Central Coast region. On Mission Plaza, the Mission San Luis



Baywood-Los Osos

California

Los Osos is an unincorporated community and a census-designated place located along the Pacific coast of San Luis Obispo County, California. The


```java
package week5;

import java.util.ArrayList;

public class Region extends Polygon {
```

Region extends Polygon

Regions have access to Polygon and Polyline methods

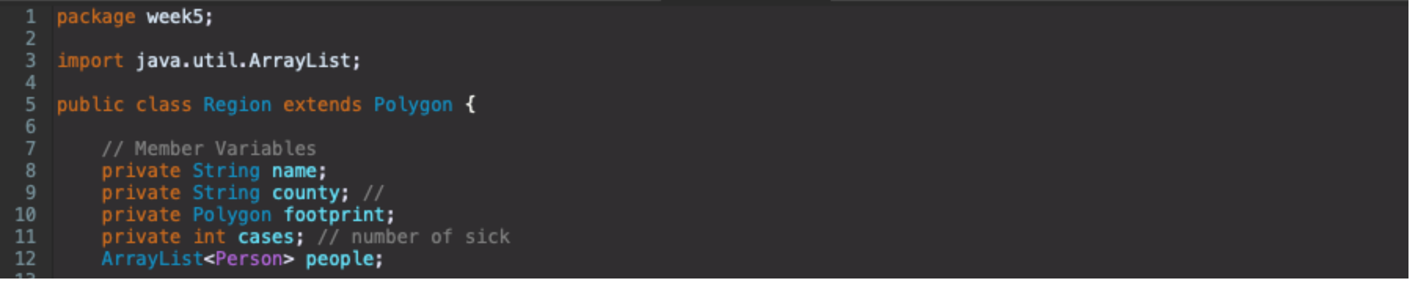
- We can ask if regions touch (Polyline.touches())
- We can generate a bounding box (Polygon.getBB()
- We can get the length (Polyline.getLength())
- We can get Point Coordinates!

How?

region.get(0).getX()

We know region extends Polygon …
which extends Polyline …
Polyline is made up of an ArrayList of Points
We delegated the ArrayList method (like get) to work on this ArrayList of POINTS

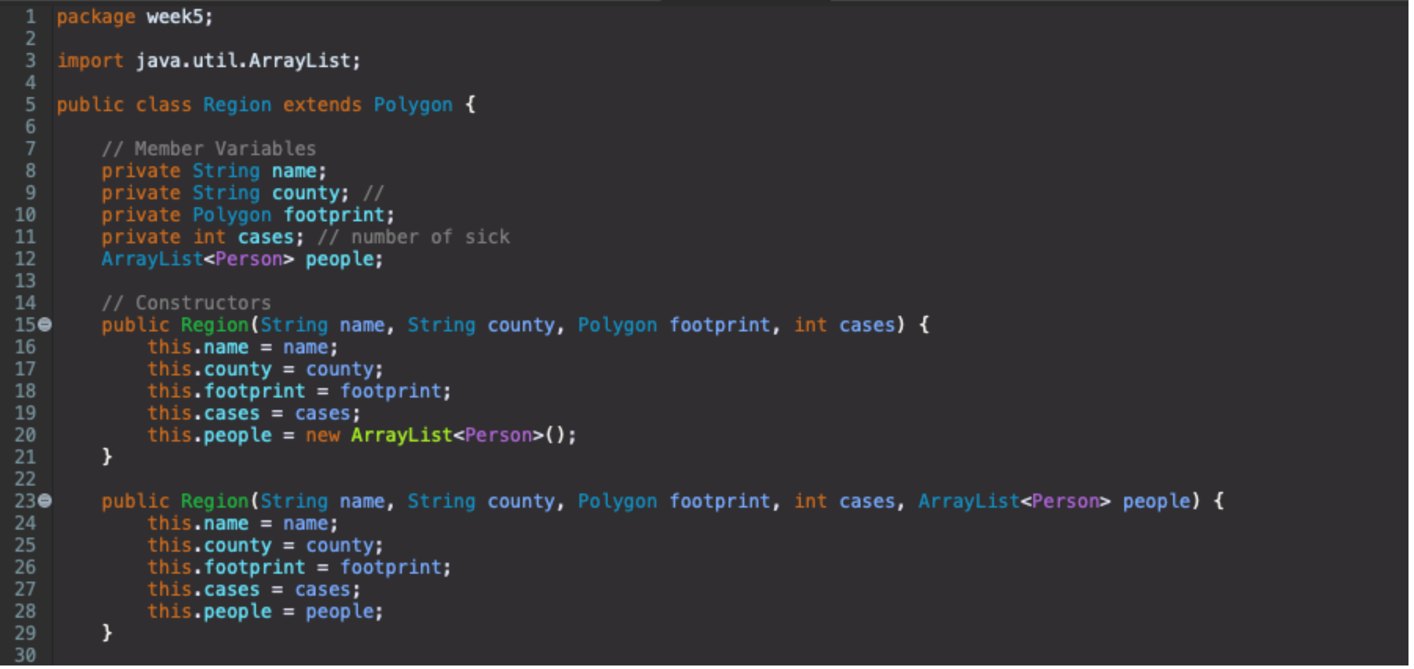
So region.get(0) returns the POINT object through the REGION → POLYGON→POLYLINE member Variable
Point objects have getters and setters to access their member variables X and Y!



```
package week5;

import java.util.ArrayList;

public class Region extends Polygon {

    // Member Variables
    private String name;
    private String county; //
    private Polygon footprint;
    private int cases; // number of sick
    ArrayList<Person> people;
```

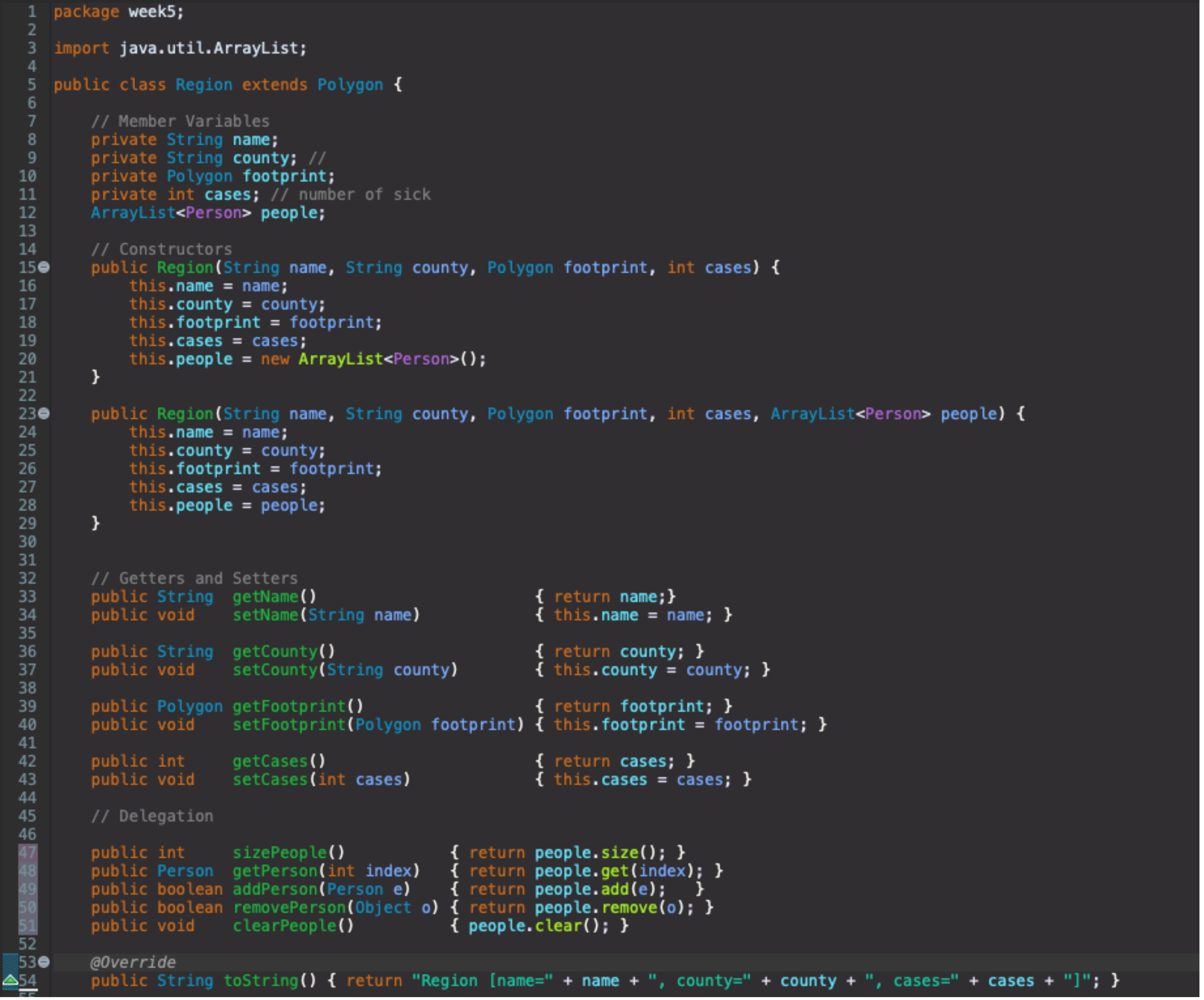
- Let’s initialize our open member variables that we know all regions have:

  Those included:
  - A name (eg String IV)
  - A county (eg String SB)
  - A footprint (eg Polygon)
  - A number of cases (eg int 100)
  - And people (ArrayList<Person>)



```
1  package week5;
2
3  import java.util.ArrayList;
4
5  public class Region extends Polygon {
6
7      // Member Variables
8      private String name;
9      private String county; //
10     private Polygon footprint;
11     private int cases; // number of sick
12     ArrayList<Person> people;
13
14     // Constructors
15     public Region(String name, String county, Polygon footprint, int cases) {
16         this.name = name;
17         this.county = county;
18         this.footprint = footprint;
19         this.cases = cases;
20         this.people = new ArrayList<Person>();
21     }
22
23     public Region(String name, String county, Polygon footprint, int cases, ArrayList<Person> people) {
24         this.name = name;
25         this.county = county;
26         this.footprint = footprint;
27         this.cases = cases;
28         this.people = people;
29     }
30
```

- Great! lets build (autogenerate!) our constructors.
- We don’t always want to deal with the people ArrayList so lets overload our constructor giving an option to include specify people (Lines 23-29) or not (Lines 15-21)…

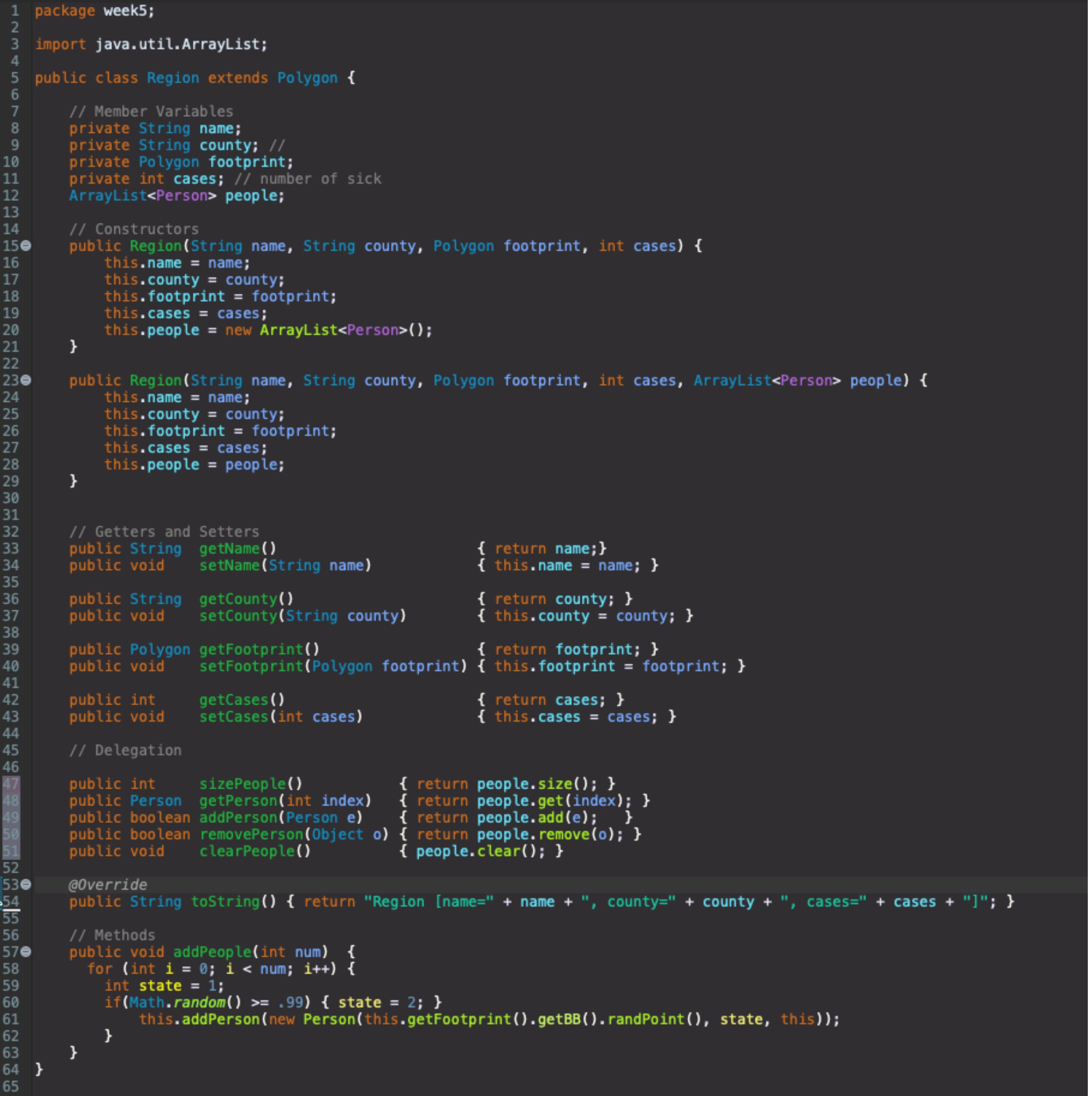


```java
package week5;

import java.util.ArrayList;

public class Region extends Polygon {

    // Member Variables
    private String name;
    private String county; //
    private Polygon footprint;
    private int cases; // number of sick
    ArrayList<Person> people;

    // Constructors
    public Region(String name, String county, Polygon footprint, int cases) {
        this.name = name;
        this.county = county;
        this.footprint = footprint;
        this.cases = cases;
        this.people = new ArrayList<Person>();
    }

    public Region(String name, String county, Polygon footprint, int cases, ArrayList<Person> people) {
        this.name = name;
        this.county = county;
        this.footprint = footprint;
        this.cases = cases;
        this.people = people;
    }


    // Getters and Setters
    public String  getName()                      { return name;}
    public void    setName(String name)           { this.name = name; }

    public String  getCounty()                    { return county; }
    public void    setCounty(String county)       { this.county = county; }

    public Polygon getFootprint()                 { return footprint; }
    public void    setFootprint(Polygon footprint) { this.footprint = footprint; }

    public int     getCases()                     { return cases; }
    public void    setCases(int cases)            { this.cases = cases; }

    // Delegation

    public int     sizePeople()             { return people.size(); }
    public Person  getPerson(int index)     { return people.get(index); }
    public boolean addPerson(Person e)      { return people.add(e);    }
    public boolean removePerson(Object o) { return people.remove(o); }
    public void    clearPeople()            { people.clear(); }

    @Override
    public String toString() { return "Region [name=" + name + ", county=" + county + ", cases=" + cases + "]"; }
```

- Auto generate:
  - Getters and Setters

- Delegate ArrayList methods to ask about the people ArrayList.
  - Note that some of these methods already apply to the inherited Polygon(Polyline) Points Array so we will modify the method name!

- Override the *toString* print method…



```java
package week5;

import java.util.ArrayList;

public class Region extends Polygon {

    // Member Variables
    private String name;
    private String county; //
    private Polygon footprint;
    private int cases; // number of sick
    ArrayList<Person> people;

    // Constructors
    public Region(String name, String county, Polygon footprint, int cases) {
        this.name = name;
        this.county = county;
        this.footprint = footprint;
        this.cases = cases;
        this.people = new ArrayList<Person>();
    }

    public Region(String name, String county, Polygon footprint, int cases, ArrayList<Person> people) {
        this.name = name;
        this.county = county;
        this.footprint = footprint;
        this.cases = cases;
        this.people = people;
    }


    // Getters and Setters
    public String  getName()                       { return name;}
    public void    setName(String name)            { this.name = name; }

    public String  getCounty()                     { return county; }
    public void    setCounty(String county)        { this.county = county; }

    public Polygon getFootprint()                  { return footprint; }
    public void    setFootprint(Polygon footprint) { this.footprint = footprint; }

    public int     getCases()                      { return cases; }
    public void    setCases(int cases)             { this.cases = cases; }

    // Delegation

    public int     sizePeople()           { return people.size(); }
    public Person  getPerson(int index)   { return people.get(index); }
    public boolean addPerson(Person e)    { return people.add(e);    }
    public boolean removePerson(Object o) { return people.remove(o); }
    public void    clearPeople()          { people.clear(); }

    @Override
    public String toString() { return "Region [name=" + name + ", county=" + county + ", cases=" + cases + "]"; }

    // Methods
    public void addPeople(int num)  {
      for (int i = 0; i < num; i++) {
        int state = 1;
        if(Math.random() >= .99) { state = 2; }
          this.addPerson(new Person(this.getFootprint().getBB().randPoint(), state, this));
        }
    }
}
```

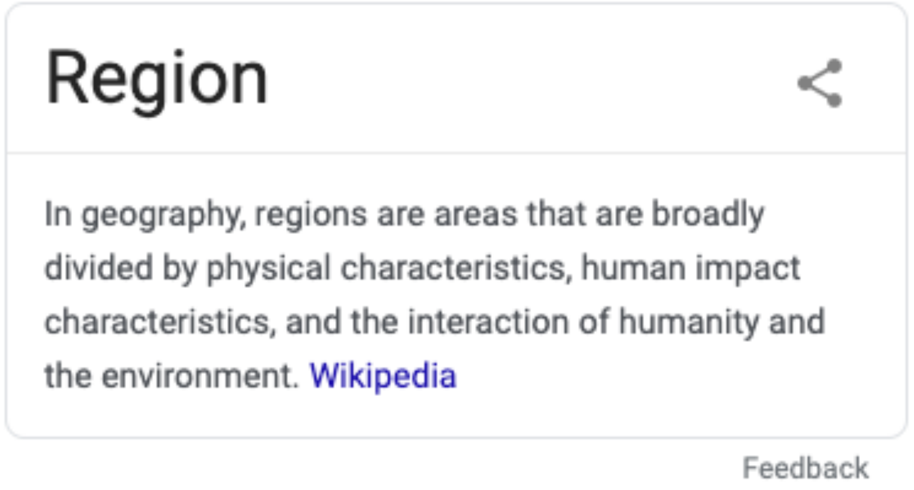
Finally, lets copy over our addPeople method from last weeks **neighborhood**…

… and modify it to work in the contexts of the **Region** Class.

Regions

All Regions have:

1. a name
2. COVID count
3. Footprint
4. County
5. People (optional)

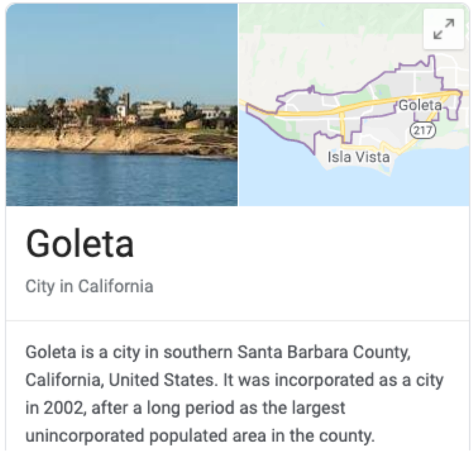


Region

In geography, regions are areas that are broadly divided by physical characteristics, human impact characteristics, and the interaction of humanity and the environment. Wikipedia

Feedback

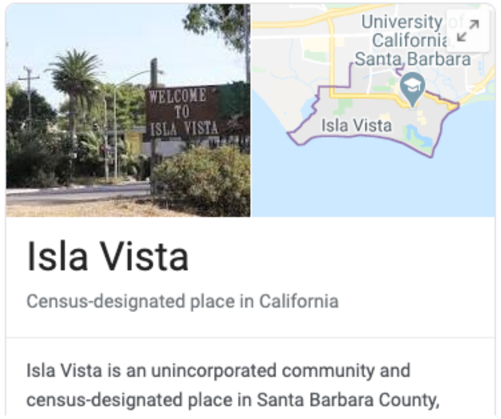
Some regions are **cities**, *some* are **unincorporated** areas:



Goleta

City in California

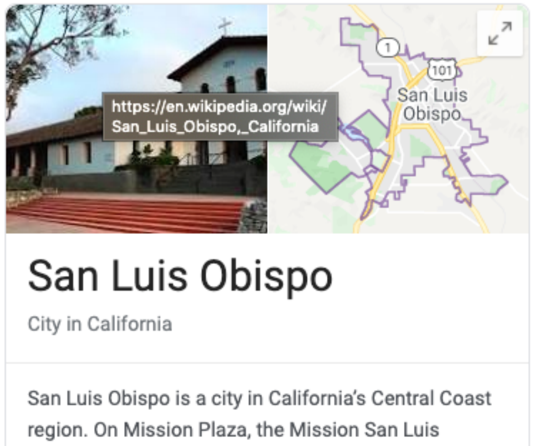
Goleta is a city in southern Santa Barbara County, California, United States. It was incorporated as a city in 2002, after a long period as the largest unincorporated populated area in the county.



Isla Vista

Census-designated place in California

Isla Vista is an unincorporated community and census-designated place in Santa Barbara County,

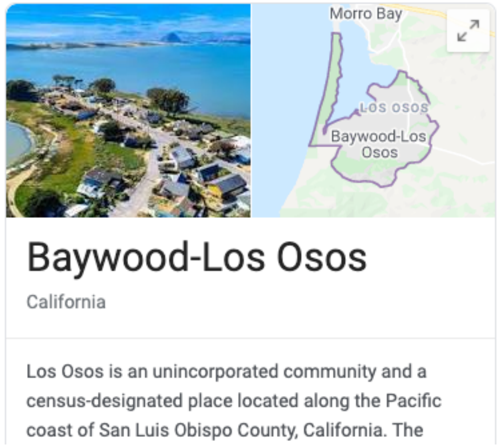


https://en.wikipedia.org/wiki/San_Luis_Obispo,_California

San Luis Obispo

City in California

San Luis Obispo is a city in California's Central Coast region. On Mission Plaza, the Mission San Luis

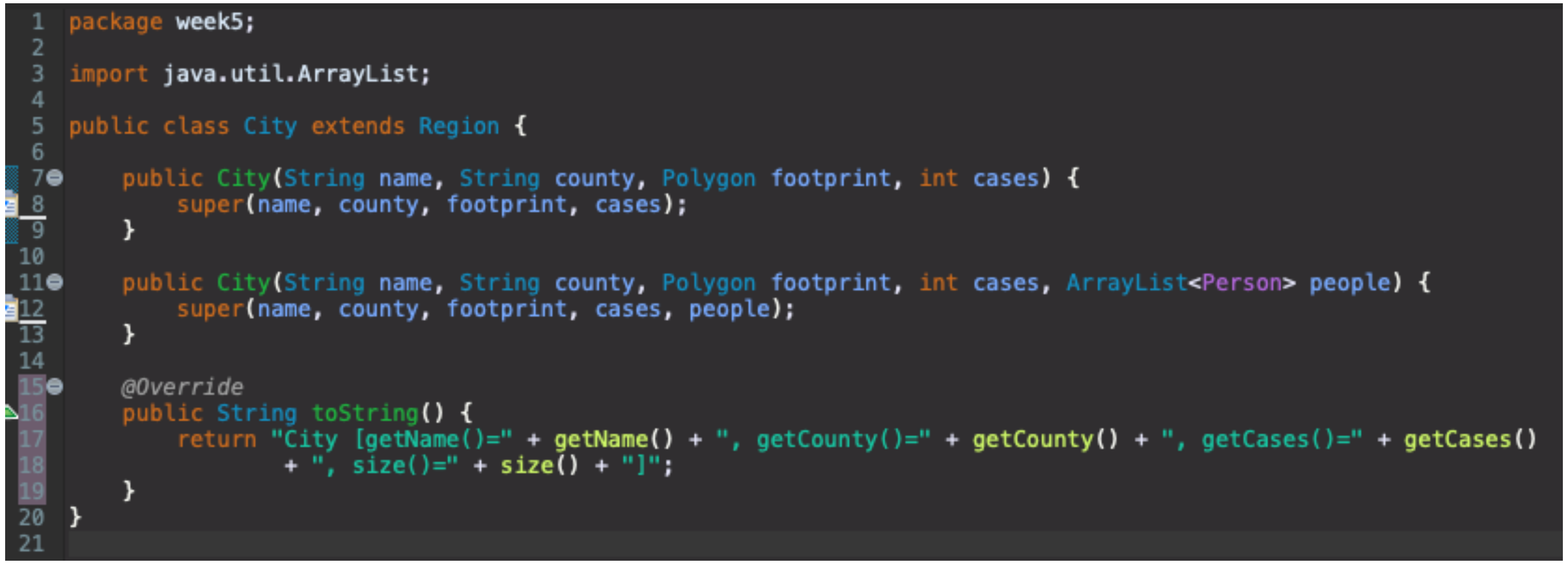


Baywood-Los Osos

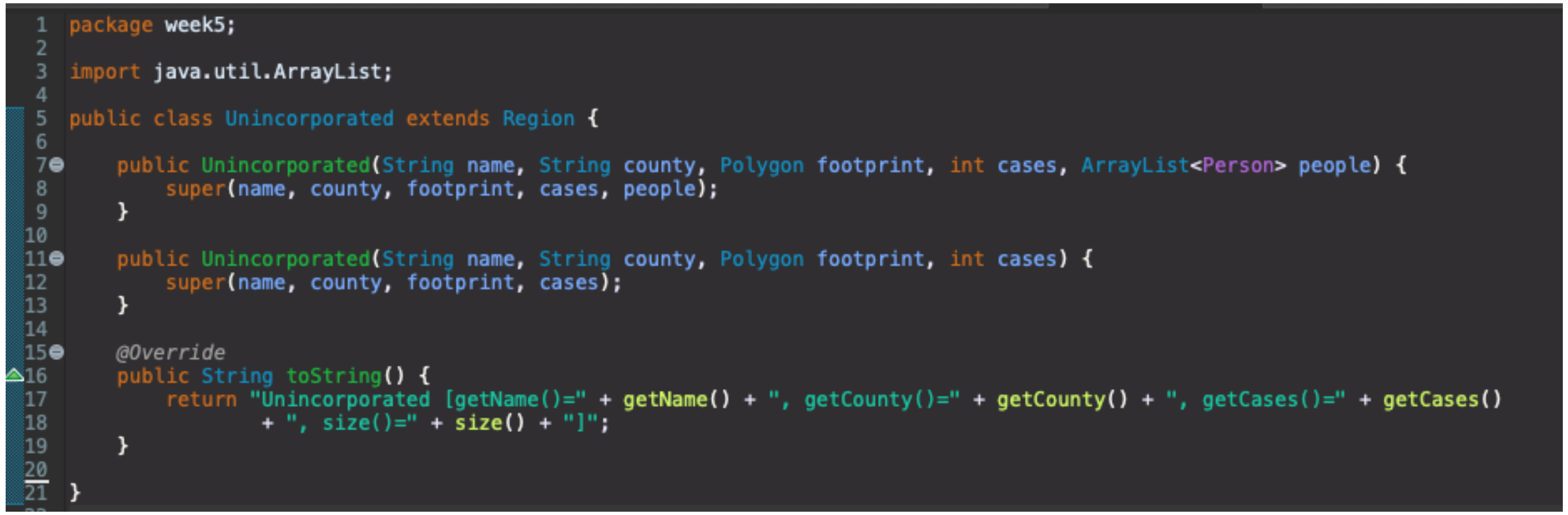
California

Los Osos is an unincorporated community and a census-designated place located along the Pacific coast of San Luis Obispo County, California. The

Let’s make Cities and Unincorporated Regions!

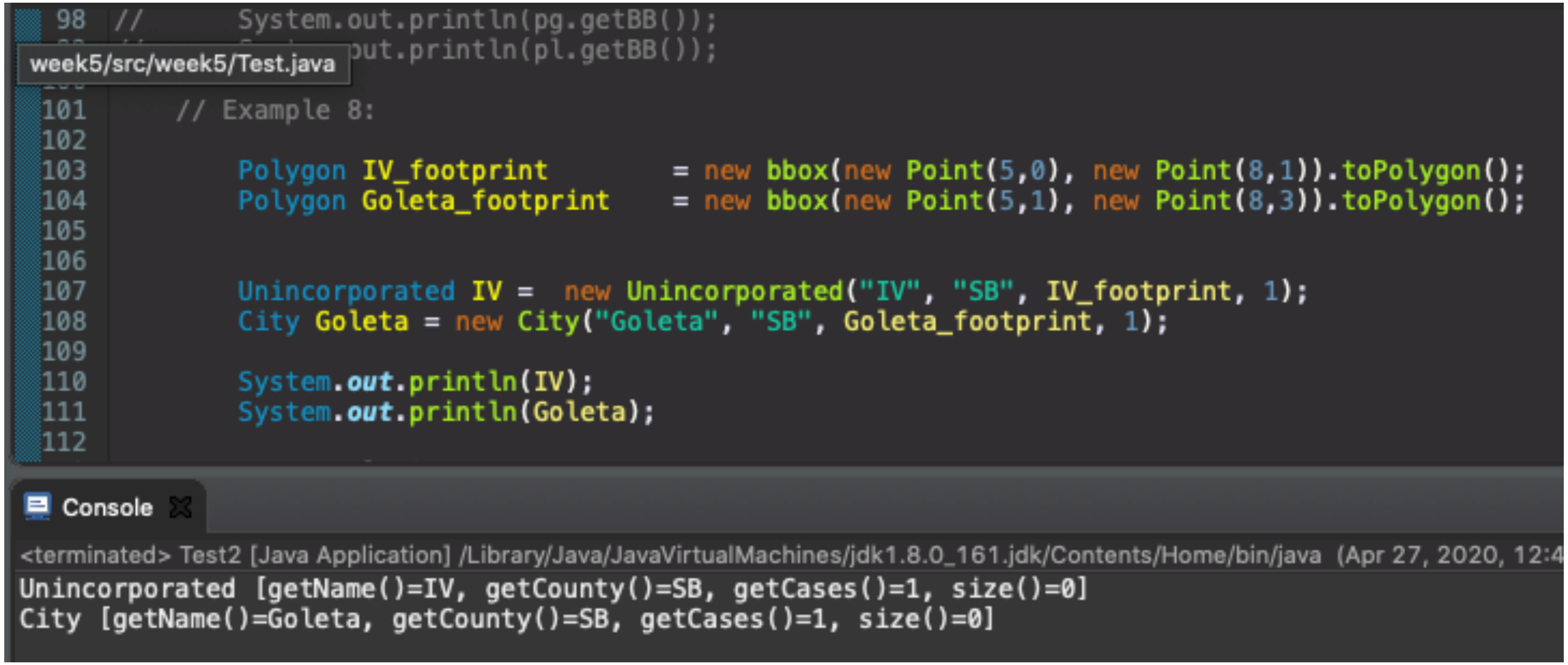


```java
package week5;

import java.util.ArrayList;

public class City extends Region {

    public City(String name, String county, Polygon footprint, int cases) {
        super(name, county, footprint, cases);
    }

    public City(String name, String county, Polygon footprint, int cases, ArrayList<Person> people) {
        super(name, county, footprint, cases, people);
    }

    @Override
    public String toString() {
        return "City [getName()=" + getName() + ", getCounty()=" + getCounty() + ", getCases()=" + getCases()
                + ", size()=" + size() + "]";
    }
}
```



```java
package week5;

import java.util.ArrayList;

public class Unincorporated extends Region {

    public Unincorporated(String name, String county, Polygon footprint, int cases, ArrayList<Person> people) {
        super(name, county, footprint, cases, people);
    }

    public Unincorporated(String name, String county, Polygon footprint, int cases) {
        super(name, county, footprint, cases);
    }

    @Override
    public String toString() {
        return "Unincorporated [getName()=" + getName() + ", getCounty()=" + getCounty() + ", getCases()=" + getCases()
                + ", size()=" + size() + "]";
    }
}
```

Example



```java
//       System.out.println(pg.getBB());
                 out.println(pl.getBB());
```

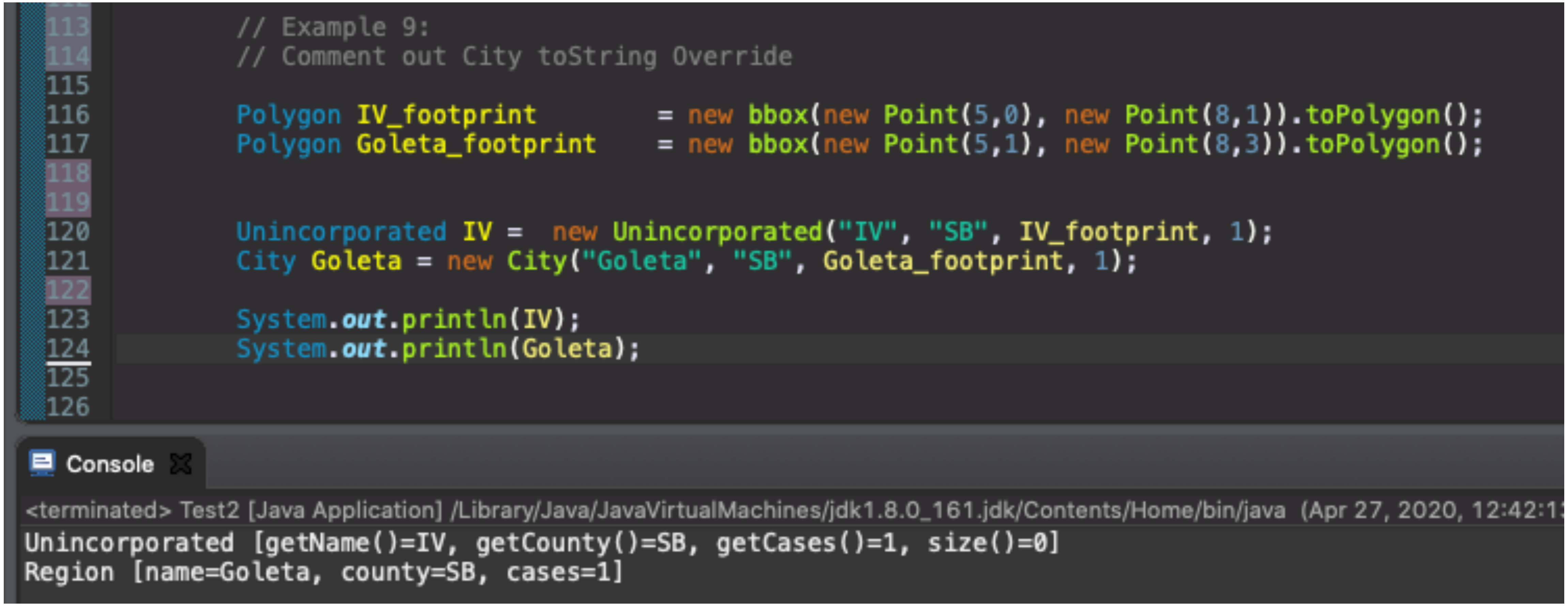
week5/src/week5/Test.java

```java
     // Example 8:

         Polygon IV_footprint         = new bbox(new Point(5,0), new Point(8,1)).toPolygon();
         Polygon Goleta_footprint     = new bbox(new Point(5,1), new Point(8,3)).toPolygon();


         Unincorporated IV =  new Unincorporated("IV", "SB", IV_footprint, 1);
         City Goleta = new City("Goleta", "SB", Goleta_footprint, 1);

         System.out.println(IV);
         System.out.println(Goleta);
```

Console

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 27, 2020, 12:4
Unincorporated [getName()=IV, getCounty()=SB, getCases()=1, size()=0]
City [getName()=Goleta, getCounty()=SB, getCases()=1, size()=0]
```

Example



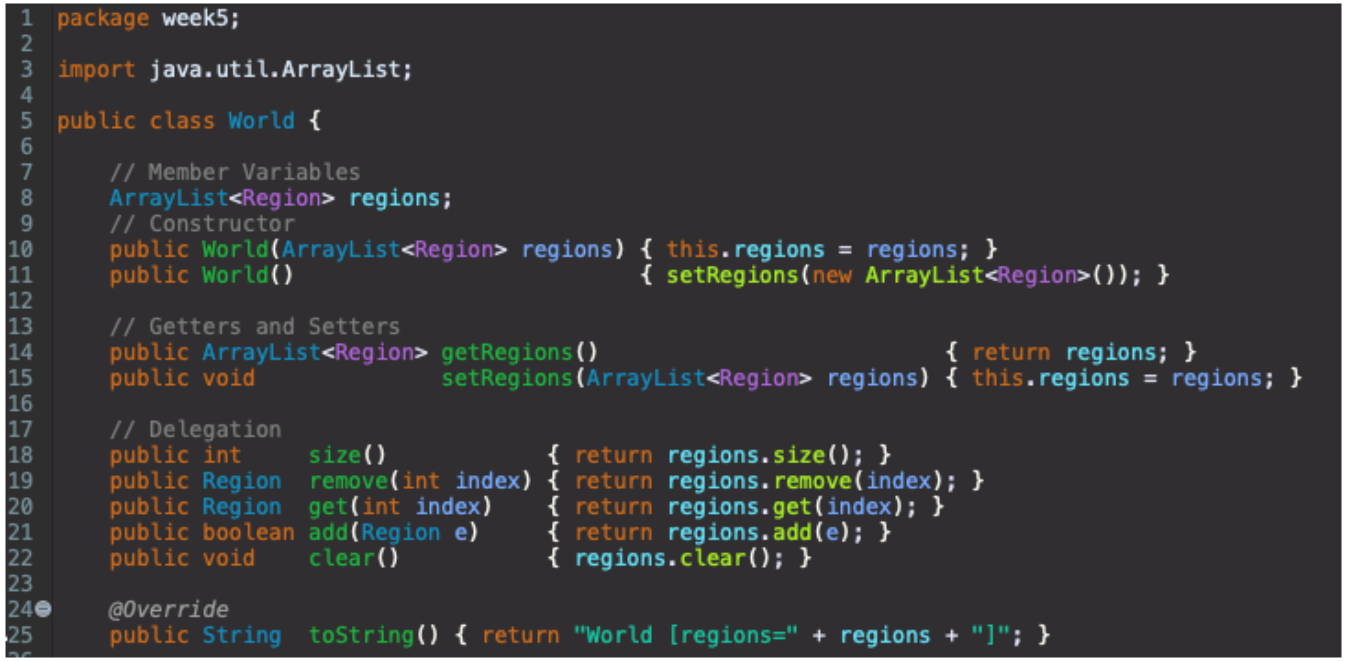
```
        // Example 9:
        // Comment out City toString Override

        Polygon IV_footprint         = new bbox(new Point(5,0), new Point(8,1)).toPolygon();
        Polygon Goleta_footprint     = new bbox(new Point(5,1), new Point(8,3)).toPolygon();


        Unincorporated IV =  new Unincorporated("IV", "SB", IV_footprint, 1);
        City Goleta = new City("Goleta", "SB", Goleta_footprint, 1);

        System.out.println(IV);
        System.out.println(Goleta);


```

Console

```
<terminated> Test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 27, 2020, 12:42:1
Unincorporated [getName()=IV, getCounty()=SB, getCases()=1, size()=0]
Region [name=Goleta, county=SB, cases=1]
```

Let’s make a *World* Class to hold all of our regions

Other then lines 8 and 11, this is all autogenerated !!



```
1  package week5;
2
3  import java.util.ArrayList;
4
5  public class World {
6
7      // Member Variables
8      ArrayList<Region> regions;
9      // Constructor
10     public World(ArrayList<Region> regions) { this.regions = regions; }
11     public World()                          { setRegions(new ArrayList<Region>()); }
12
13     // Getters and Setters
14     public ArrayList<Region> getRegions()                      { return regions; }
15     public void              setRegions(ArrayList<Region> regions) { this.regions = regions; }
16
17     // Delegation
18     public int     size()            { return regions.size(); }
19     public Region  remove(int index) { return regions.remove(index); }
20     public Region  get(int index)    { return regions.get(index); }
21     public boolean add(Region e)     { return regions.add(e); }
22     public void    clear()           { regions.clear(); }
23
24     @Override
25     public String  toString() { return "World [regions=" + regions + "]"; }
```

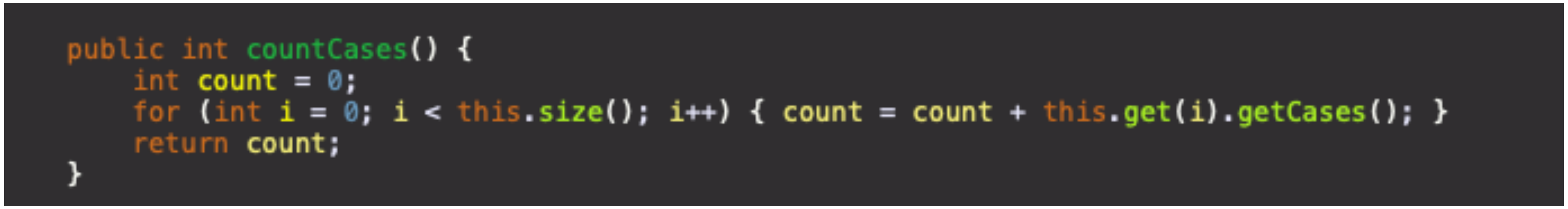
Check in!

- You should have:
  1. Point class (with equals override)
  2. Polyline class with getLength and touches method
     - Polygon class with getBB class
  3. Region class with addPeople class
     - City class
     - Unincorporated class
  4. World class

All classes should have the needed member variables, getters & setters, delegated methods, and toString overrides

Count Cases

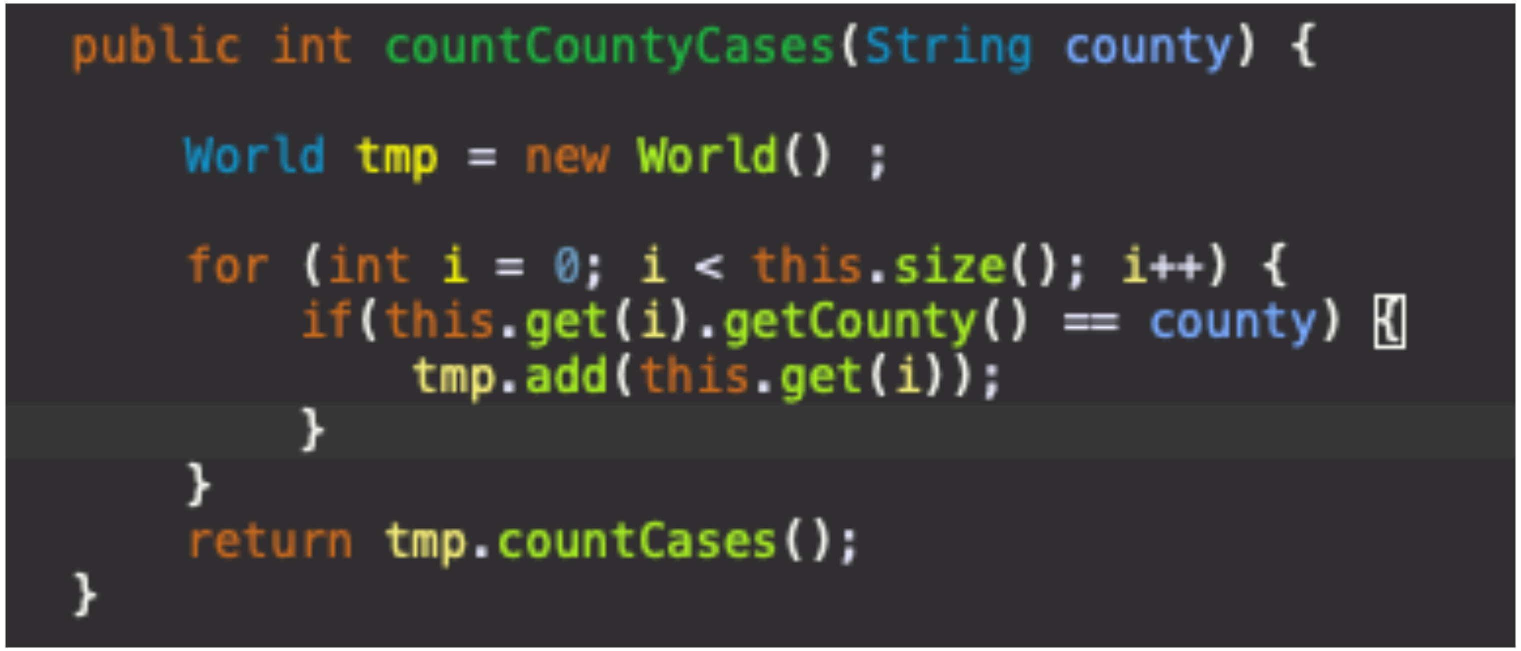
Lets add a method that lets us count all cases in a World Object:



```
public int countCases() {
    int count = 0;
    for (int i = 0; i < this.size(); i++) { count = count + this.get(i).getCases(); }
    return count;
}
```

Count County Cases

Let’s add a method that lets us count all cases in a World Object that match a criteria: countyName == *input*



```
public int countCountyCases(String county) {

    World tmp = new World() ;

    for (int i = 0; i < this.size(); i++) {
        if(this.get(i).getCounty() == county) {
            tmp.add(this.get(i));
        }
    }
    return tmp.countCases();
}
```

- Logic: Create an empty world object using the “default” constructor
- Loop over all counties in the world that the method is applied to
- add all counties that meet the county name constraint to the temporary world
- Apply the countCases world method to the temporary world
- Remember that tmp ”dies” when the scope of the function ends!

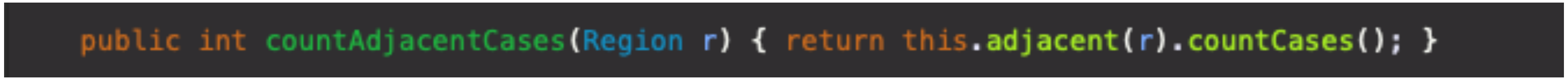
Calculate Adjacency

Build a method that calculates the adjacent regions and returns a new world object   Fill in the <…>

```
public World adjacent(Region r) {

        World tmp = new World();  //initialize a new empty world

        for (int i = 0; i < this.size(); i++) { // loop over World that this method is applied too
            if(r. <…>.touches(<…>)) {      // apply logic to see if the region (i) touches the input region
                tmp.add(<…>);             //  if it does (TRUE) add the touching region to the tmp object
            }
        }
        return tmp; // return the tmp object
}
```

Count Adjacency Cases

Build a method that counts the cases in adjacent regions using already defined methods



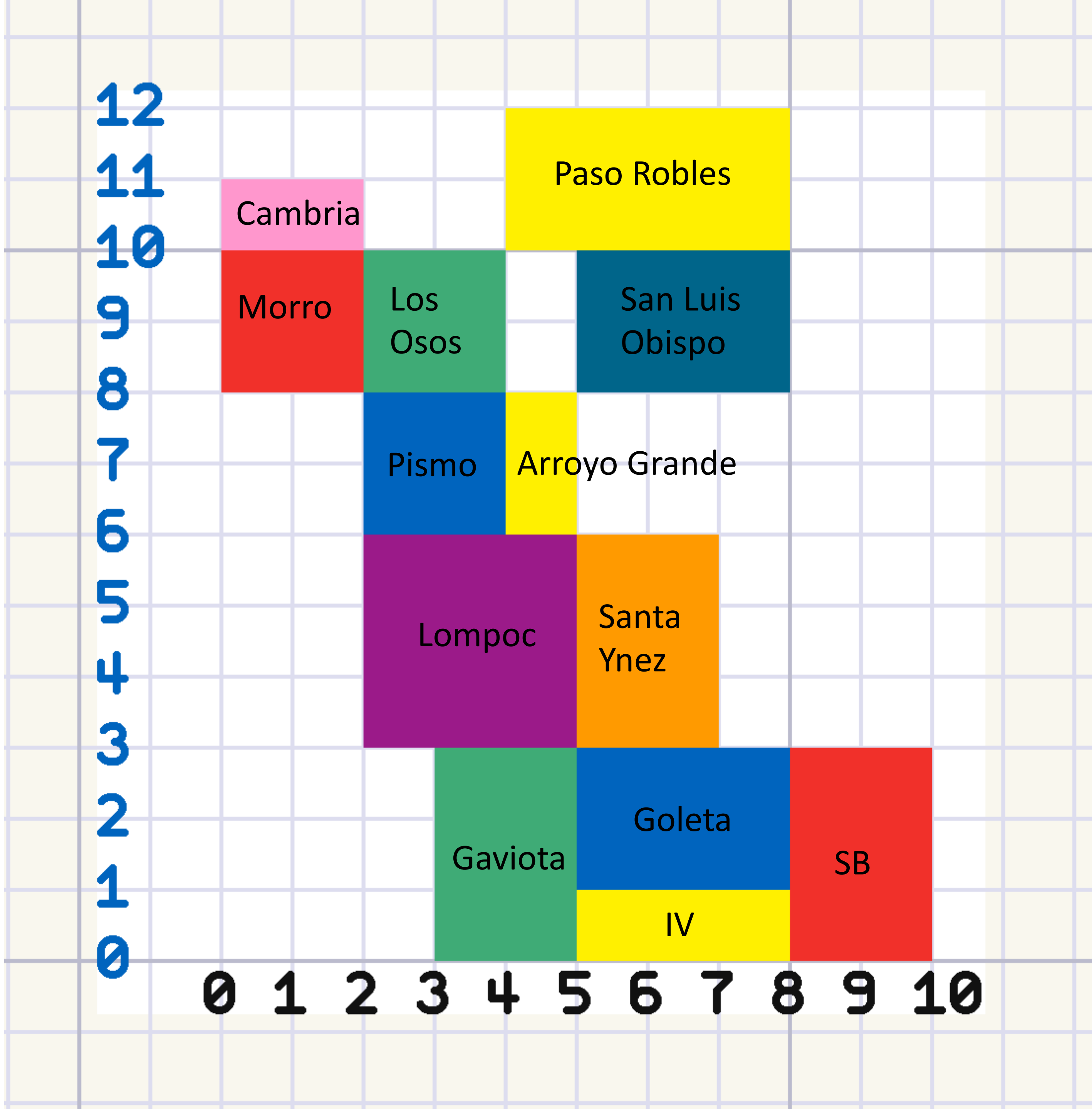
```
public int countAdjacentCases(Region r) { return this.adjacent(r).countCases(); }
```

Homework Hints:

At minimum make the following regions:

Type	Name	Cases
Unincorporated	IV	250
city	Goleta	400
city	SB	300
city	Santa Ynez	100
city	SLO	250
city	Arroyo Grande	150
Unincorporated	Los Osos	50

1. Add them to a world called CC (central coast)
2. Count cases in the CC (should be 1500)
3. Add 20,000 people to IV
4. Print the size of IV Population
5. Print the cases in IV (250)
6. Print the adjacent Regions to IV
   (City Goleta, Uni. IV, and City SB)
7. Print the adjacent Regions to Santa Ynez
   (City Goleta, city Santa Ynez, city AG)
8. Count the adjacent cases to IV (950)
9. Count the cases in SLO county (450)
10. Count the cases in SB county (1050)


12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10
Paso Robles
Cambria
Morro
Los Osos
San Luis Obispo
Pismo
Arroyo Grande
Lompoc
Santa Ynez
Gaviota
Goleta
SB
IV