GEOG 178/258
Week 4:

ArrayLists, Delegation, and Randomization

mike johnson

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
int v = 10;

if(v <= 5) {
    System.out.print("Less then or equal to 5");
} else {
    System.out.print("Greater then 5");
}
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
int v = 10;
for (int i = 0; i < 50; i++) {
    v = v + i;
}

System.out.print(v);
// v = 1235
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

•The **main**() **method** is the entry point into the application.
•The signature of the **method** is always: public static void **main**(String[] args)
•Command-line arguments are passed through the args parameter, which is an array of String s. **We do not deal with these!!**

```
5  public class test {
6
7      public static void main(String[] args) {
8
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate) (this vs. that)
  - Getters Setters (auto-generate)
  - Methods
  - Access

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
3  public class point {
4
5      private double x, y;
6
7      public point(double x, double y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     public double getX() { return x; }
13     public double getY() { return y; }
14
15     public void setX(double x) { this.x = x; }
16     public void setY(double y) { this.y = y; }
17
18     public double dist(double x, double y) {
19         return Math.sqrt(Math.pow(this.x - x, 2) +
20                 Math.pow((this.y - y), 2));
21     }
22
23     public double dist(point p) {
24         return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
25                 Math.pow((this.y - p.getY()), 2));
26     }
27
28     public void toWKT() {
29        System.out.println("POINT" + this.getX() + " " + this.getY());
30     }
31
32 }
33
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate) (this vs. that)
  - Getters Setters (auto-generate)
  - Methods
  - Access

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
3  public class point {
4
5      private double x, y;
6
7      public point(double x, double y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     public double getX() { return x; }
13     public double getY() { return y; }
14
15     public void setX(double x) { this.x = x; }
16     public void setY(double y) { this.y = y; }
17
18     public double dist(double x, double y) {
19         return Math.sqrt(Math.pow(this.x - x, 2) +
20                 Math.pow((this.y - y), 2));
21     }
22
23     public double dist(point p) {
24         return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
25                 Math.pow((this.y - p.getY()), 2));
26     }
27
28     public void toWKT() {
29        System.out.println("POINT" + this.getX() + " " + this.getY());
30     }
31
32 }
33
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate) (this vs. that)
  - Getters Setters (auto-generate)
  - Methods
  - Access

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
3  public class point {
4
5      private double x, y;
6
7      public point(double x, double y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     public double getX() { return x; }
13     public double getY() { return y; }
14
15     public void setX(double x) { this.x = x; }
16     public void setY(double y) { this.y = y; }
17
18     public double dist(double x, double y) {
19         return Math.sqrt(Math.pow(this.x - x, 2) +
20                 Math.pow((this.y - y), 2));
21     }
22
23     public double dist(point p) {
24         return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
25                 Math.pow((this.y - p.getY()), 2));
26     }
27
28     public void toWKT() {
29        System.out.println("POINT" + this.getX() + " " + this.getY());
30     }
31
32 }
33
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate) (this vs. that)
  - Getters Setters (auto-generate)
  - Methods (duplication)
  - Access

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
3  public class point {
4
5      private double x, y;
6
7      public point(double x, double y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     public double getX() { return x; }
13     public double getY() { return y; }
14
15     public void setX(double x) { this.x = x; }
16     public void setY(double y) { this.y = y; }
17
18     public double dist(double x, double y) {
19         return Math.sqrt(Math.pow(this.x - x, 2) +
20                 Math.pow((this.y - y), 2));
21     }
22
23     public double dist(point p) {
24         return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
25                 Math.pow((this.y - p.getY()), 2));
26     }
27
28     public void toWKT() {
29        System.out.println("POINT" + this.getX() + " " + this.getY());
30     }
31
32 }
33
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate) (this vs. that)
  - Getters Setters (auto-generate)
  - Methods (duplication)
  - Access

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
public class point {

    private double x, y;

    public point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() { return x; }
    public double getY() { return y; }

    public void setX(double x) { this.x = x; }
    public void setY(double y) { this.y = y; }

    public double dist(double x, double y) {
        return Math.sqrt(Math.pow(this.x - x, 2) +
                Math.pow((this.y - y), 2));
    }

    public double dist(point p) {
        return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
                Math.pow((this.y - p.getY()), 2));
    }

    public void toWKT() {
        System.out.println("POINT" + this.getX() + " " + this.getY());
    }

}
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

```
3  public class point {
4
5      private double x, y;
6
7      public point(double x, double y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     public double getX() { return x; }
13     public double getY() { return y; }
14
15     public void setX(double x) { this.x = x; }
16     public void setY(double y) { this.y = y; }
17
18     public double dist(double x, double y) {
19         return Math.sqrt(Math.pow(this.x - x, 2) +
20                 Math.pow((this.y - y), 2));
21     }
22
23     public double dist(point p) {
24         return Math.sqrt(Math.pow(this.x - p.getX(), 2) +
25                 Math.pow((this.y - p.getY()), 2));
26     }
27
28     public void toWKT() {
29        System.out.println("POINT" + this.getX() + " " + this.getY());
30     }
31
32 }
33
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bounding Box

- Arrays
- ArrayLists (import package)

```
3  public class bbox {
4
5      //Attributes
6      double xmax, xmin, ymax, ymin;
7
8      //Constructors
9      public bbox(point p1, point p2) {
15
16     public double getXmax() { return xmax; }
17     public void   setXmax(double xmax) { this.xmax = xmax; }
18
19     public double getXmin() { return xmin; }
20     public void   setXmin(double xmin) { this.xmin = xmin; }
21
22     public double getYmax() { return ymax; }
23     public void   setYmax(double ymax) { this.ymax = ymax; }
24
25     public double getYmin() { return ymin; }
26     public void   setYmin(double ymin) { this.ymin = ymin; }
27
28     public boolean isInside(point p) {
29
30         return  p.getX()>=this.xmin && p.getX()<=this.xmax &&
31                 p.getY()<=this.ymin && p.getY()>=this.ymax  ;
32     }
33
34 }
```

Recap

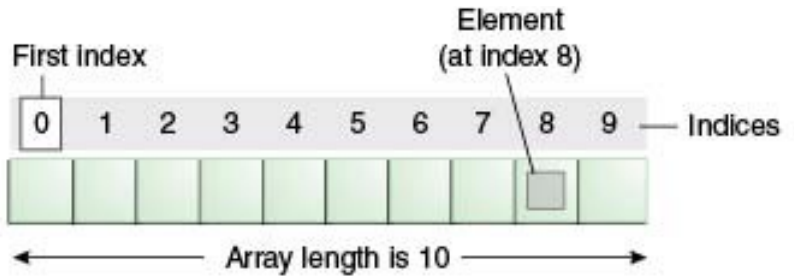
- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

Zero Index, Same Type, Fixed Length



An array of 10 elements.

```
1  package week3;
2
3  public class Arrays {
4      public static void main(String[] args) {
5
6          double[] UCSB = new double[2];
7
8          UCSB[0] = 34.41;
9          UCSB[1] = -119.84;
10
11         System.out.print("The Latitude of UCSB is " + UCSB[0] + "." +
12                          "\n\nThe Longitude of UCSB is "+ UCSB[1] + "." +
13                  "\n\nUCSB is located at POINT (" + UCSB + ")"
14              );
15
16     }
17
18 }
```

Recap

- If statement
- For loop
- Main method

- Classes/objects
  - Member attributes
  - Constructors (auto-generate)
  - Getters Setters (auto-generate)
  - Methods

- Point
- Bbounding Box

- Arrays
- ArrayLists (import package)

Zero Index, Same Type, Flexible Length

An ArrayList is a re-sizable **array**, also called a dynamic **array**. It grows its size to accommodate new elements and shrinks the size when the elements are removed. ArrayList internally uses an **array** to store the elements. Just like **arrays**, It allows you to retrieve the elements by their index.

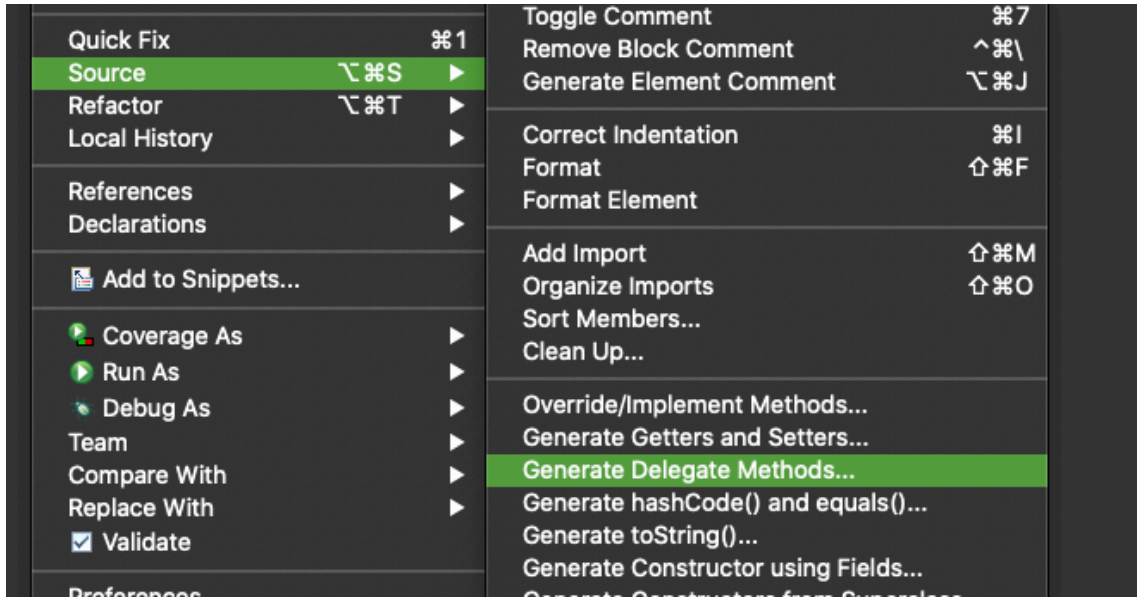
import java.util.ArrayList; *// Must be imported!*

```
ArrayList<point> polyline = new ArrayList<point>();
double[] UCSB = new double[2];
```

Established "operations" or methods that can be done on the ArrayList class

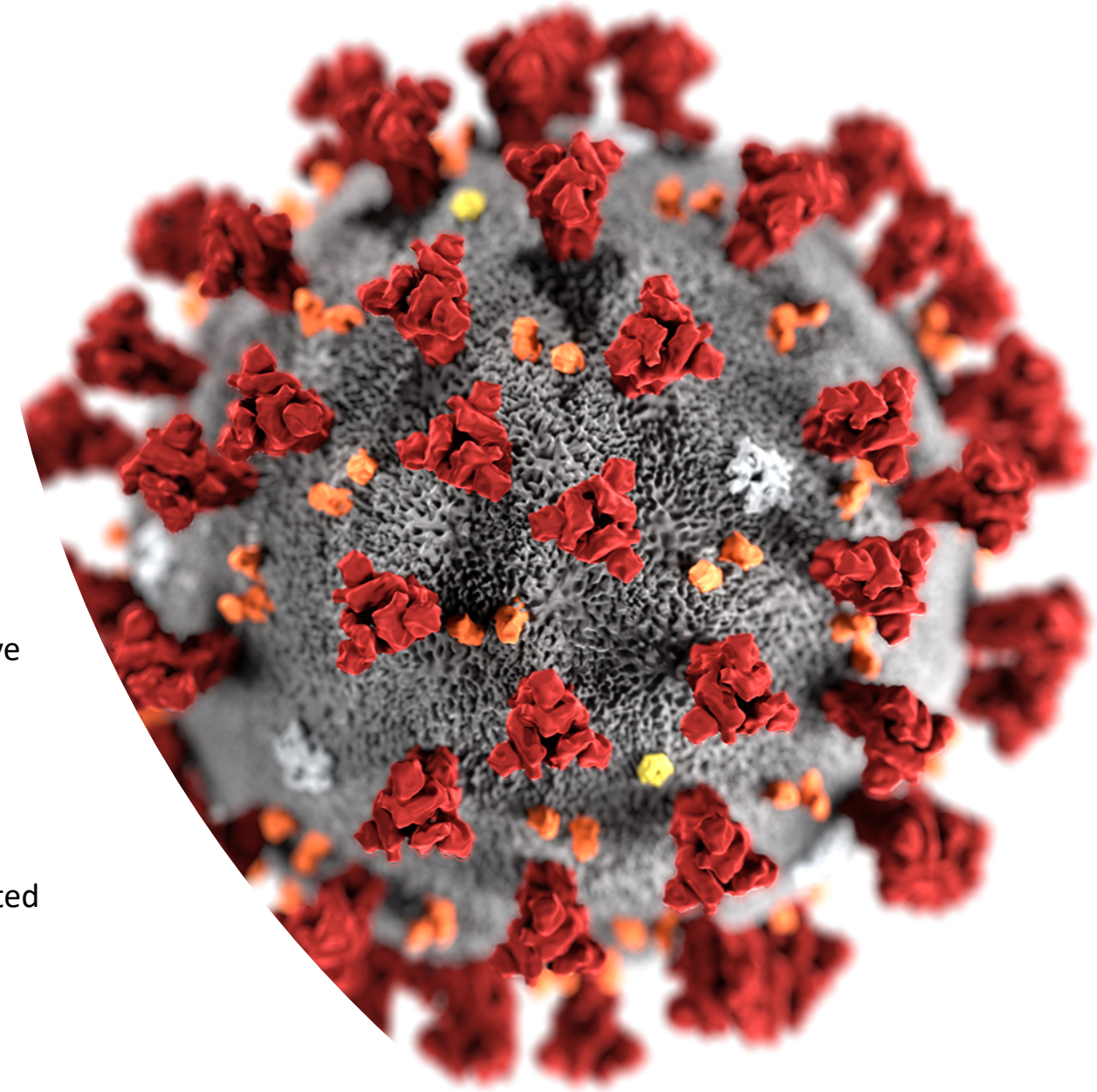
Delegation

- 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();
    }
}
```

COVID Example

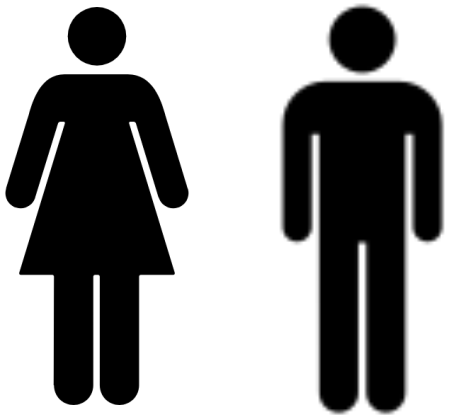


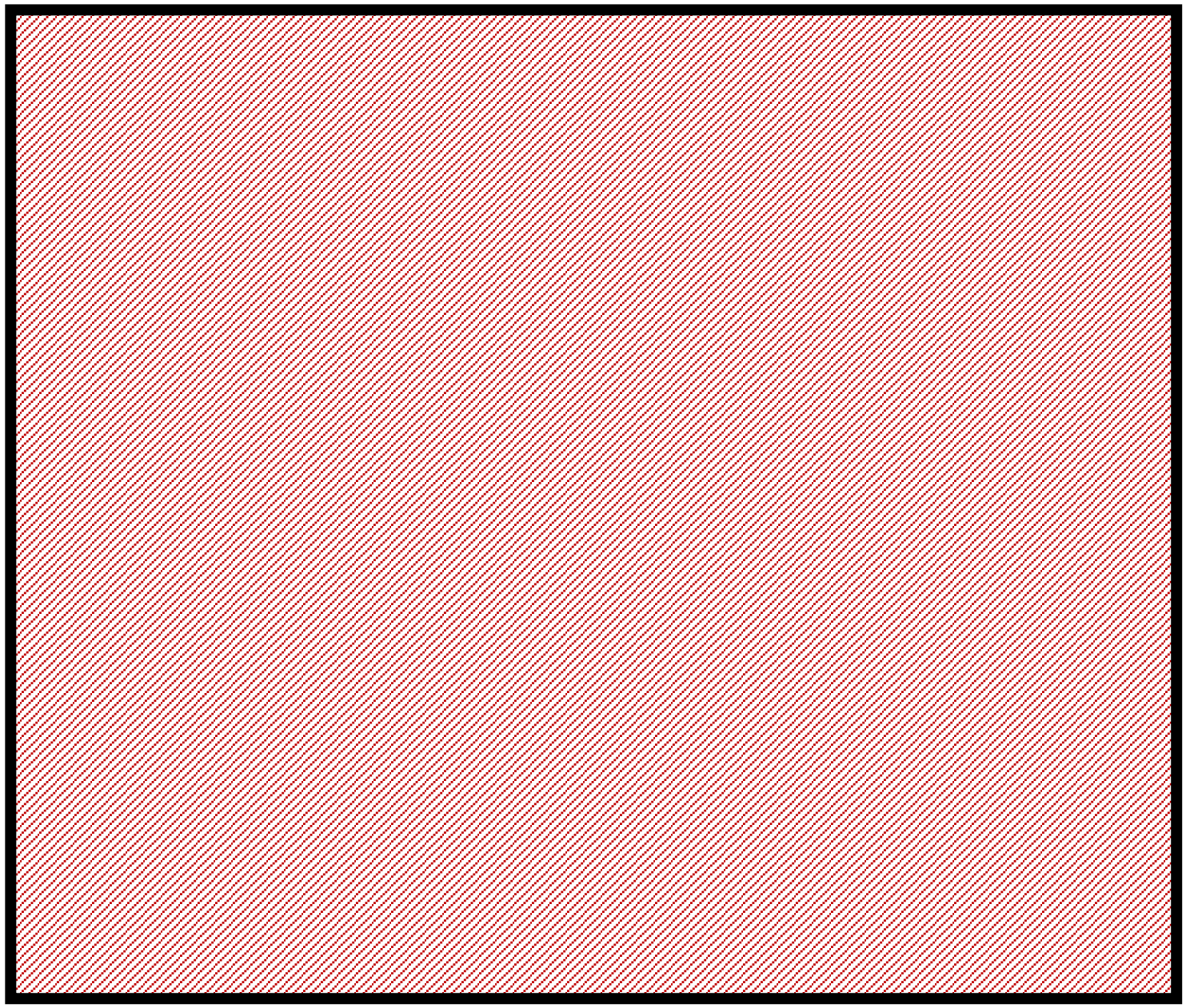
- We want to model the spread of infection between **people** in **neighborhoods**
  - We will focus on **1 neighborhood**
  - **People** occupy some point in the neighborhood and cannot move
  - **People** are either infected (state = 1) or not-infected (state = 2)
  - At the beginning only 1% of the population is infected.
  - Premise:

    Unaffected people within 6 feet (Cartesian space) of an infected person become infected

Model

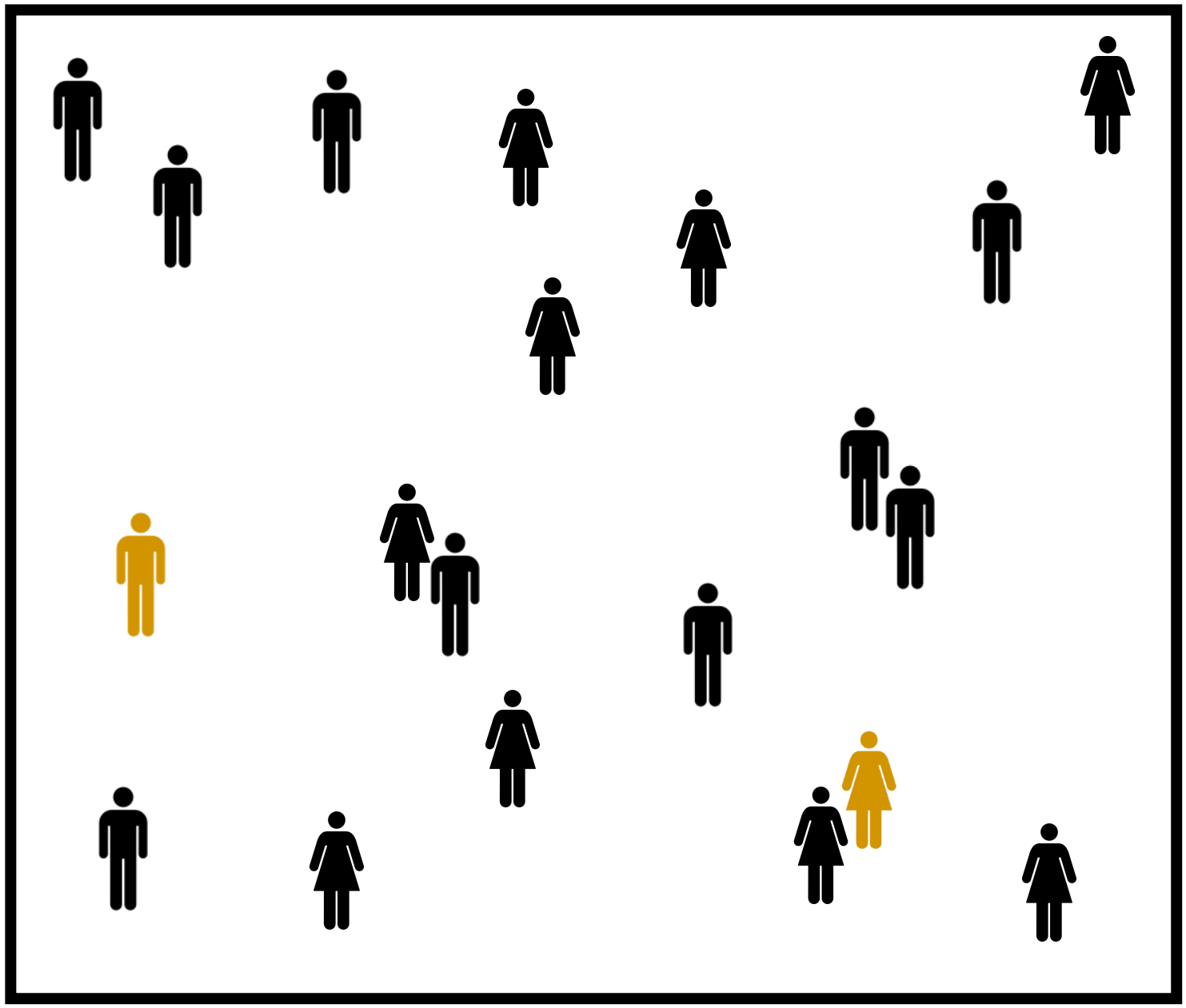
- Neighborhood
- People





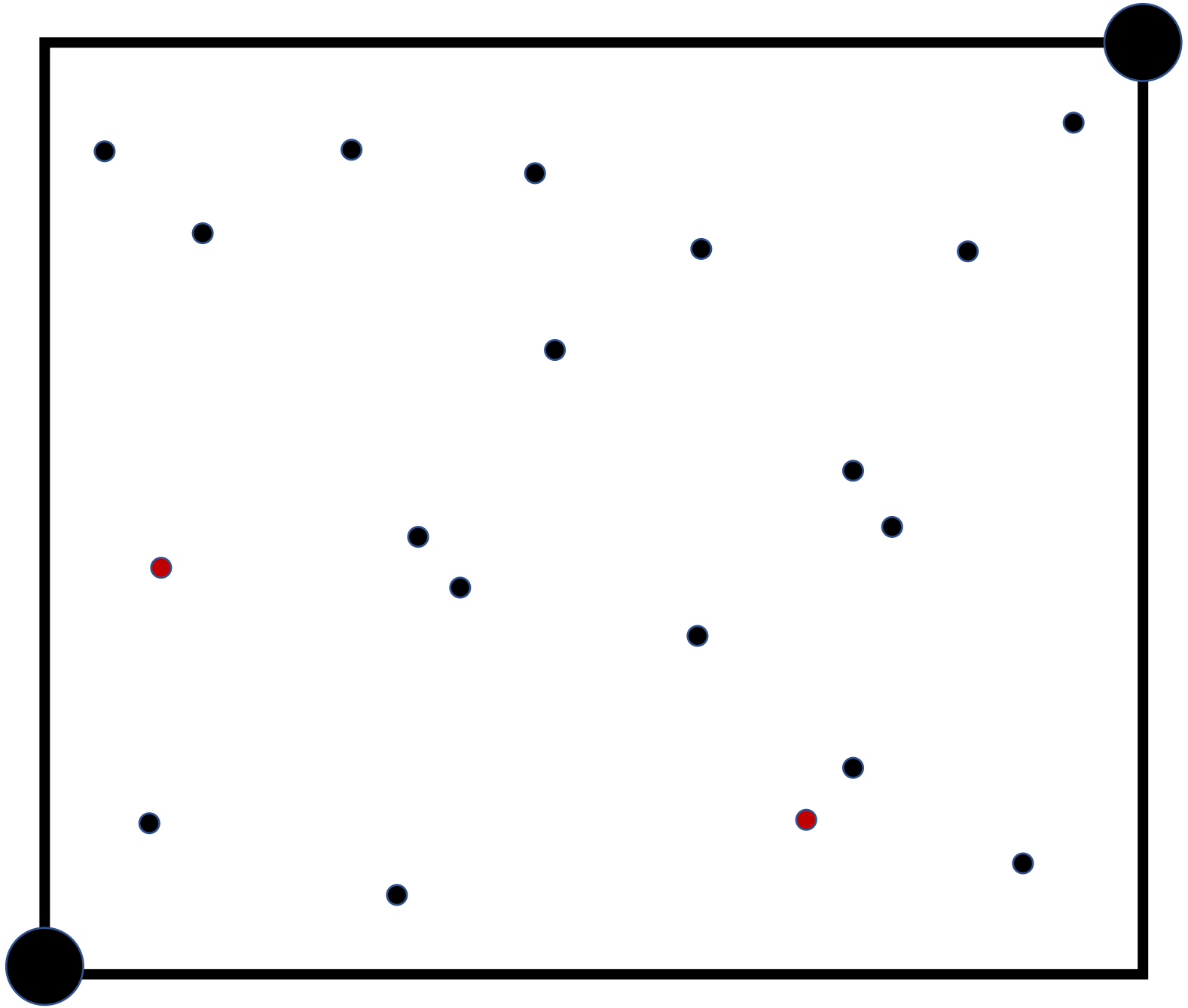
Model

- Neighborhood
  - (Area)
- People (n = 20)
  - (location, infected-state)

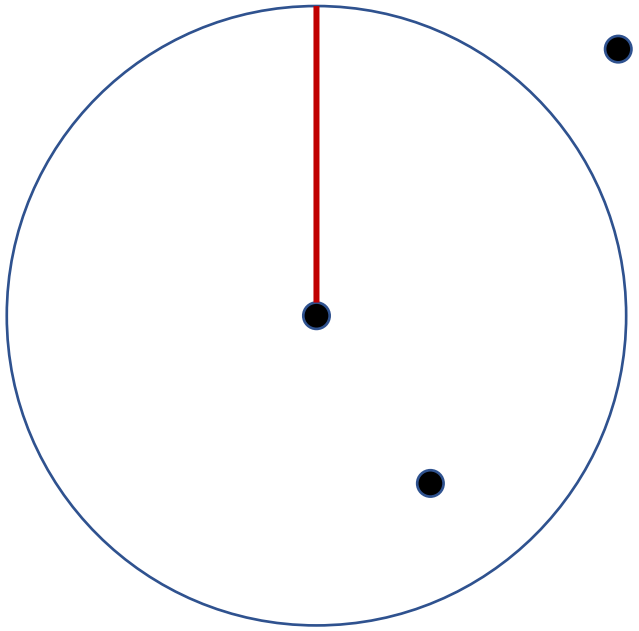


Model

- Neighborhood
  - (Area) -- >
  - bounding box!
- People
  - (location, infected-state) -->
  - point!



What else do we need?



- **Is a person within 6 feet of another?**
  - Choice: Is this a person method or a point method?

- **Randomness (model initialization)!**
  - Where people are
  - Are they infected?

Random Numbers in Java

Math.random()

- Gives you a random double between 0 and 1 (e.g. .6503939429)
- Includes 0, Excludes 1 !!

- double rand = Math.rand();
- Each time you run it, you get a different number between 0-1 (1 not included)

Random Numbers between a range?

- Lets say we want random numbers between 5-10

- Math.random() * (max – min)
  (rand * 5) → returns the value in range of 0-5 where 5 is **not** included
  *(0 * 5) = 0*
  *(.9999 * 5) = 4.9995*

- Math.random() * (max – min) + min (exclusive)
  (rand * 5) + 5 → returns the value in range of 5-10 where 10 is **not** included
  *(0 * 5) + 5 = 5*
  *(.9999 * 5) + 5 = 9.9995*

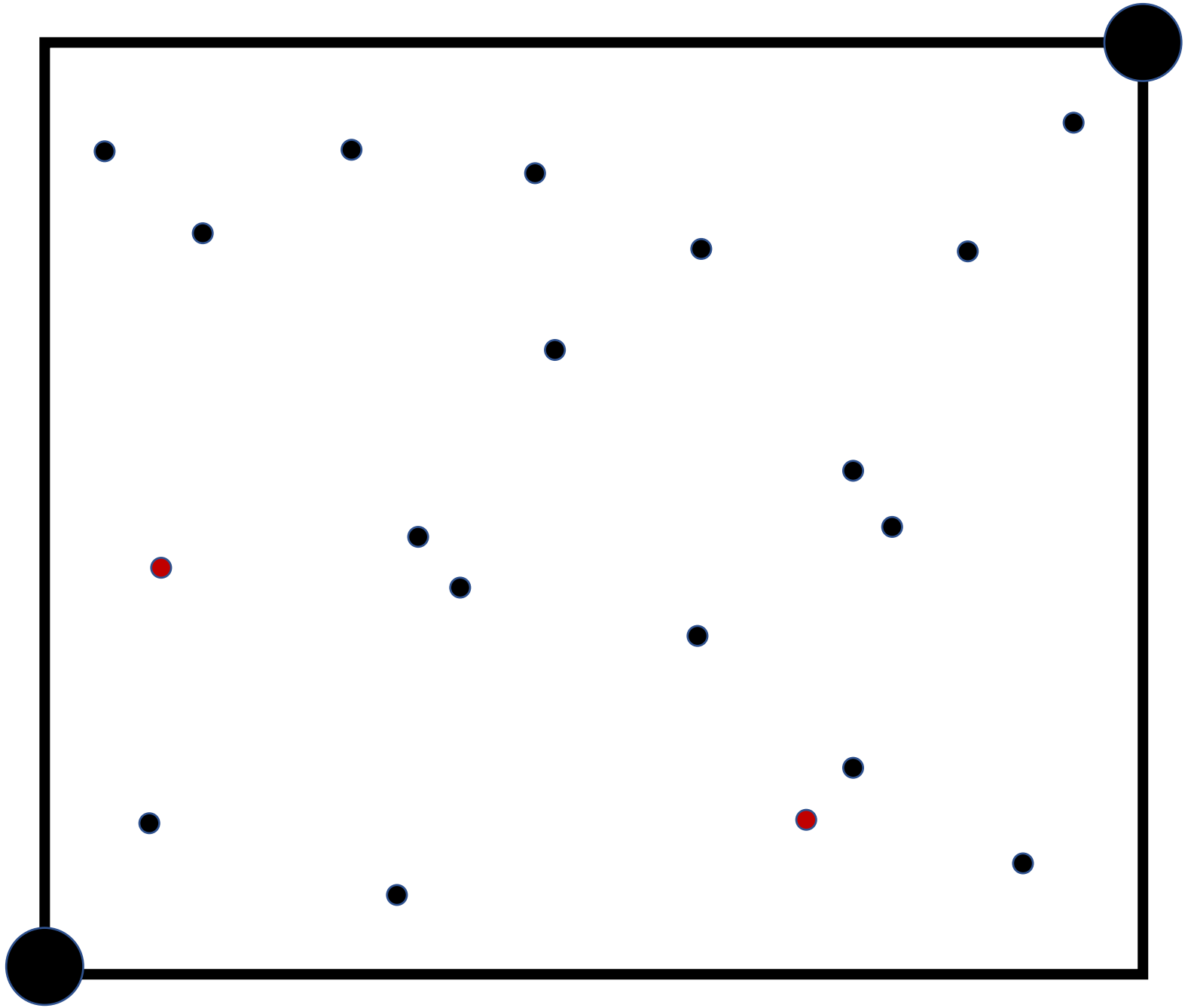
- Math.random() * (max – min + 1) + min (inclusive)
  (rand * 6) + 5 → returns the value in range of 5-10 where 10 **is** included
  *(0 * 6) + 5 = 5*
  *(.9999 * 6) + 5 = 10.9994*

Random Int (introduction to casting data type)

- Here we need to cast a double to an int
- Casting a double to an integer is the equivalent to floor()
- (int) 6.999999999999 = 6

- Int rand = (int) (Math.random() * ((max – min + 1) + min));

Lets work it out



Start a new project.

- Copy in your point and bbox class from last week
- Create 3 new classes
  - Person
  - Neighborhood
  - Test (add a main method here!)

  - Example code that follows with be color coded (border) according to these.

Make a Person class

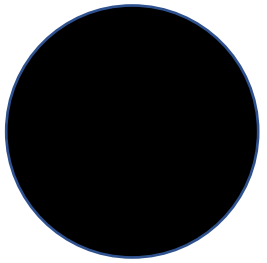
- What defines a person in the contexts of this model?
- Do attributes like hair color, eye color, name matter?

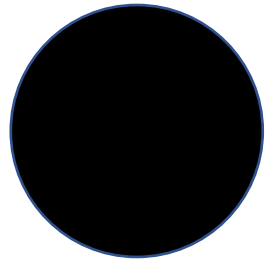
THIS

THIS

or

THIS





Attributes of person

```
3 public class person {
4
5     //attribute
6         private point location;
7         private int   state; // 1=s, 2=i, 3=r.
```

Line 3: We create a public class called person
Line 6: All person(s) have a point attribute, called location, that is only accessible to the class
Line 7: All person(s) have an integer attribute, called location, that is only accessible to the class

Constructors create specific *objects* of class *person*

```
 3 public class person {
 4
 5     //attribute
 6         private point location;
 7         private int   state; // 1=s, 2=i, 3=r.
 8
 9         public person(point location, int state) {
10             super();
11             this.location = location;
12             this.state = state;
13         }
```

Line 9: this is the constructor because the name matches the class. It is public and requires the user to supply a point an integer
Line 10: Ignore this for now
Line 11: Assign the input point to the location of **THIS** specific person being created any time the constructor is applied
Line 12: Assign the input integer to the state of **THIS** specific person being created any time the constructor is applied

Provide Getters and setters for member variables

```
15          public point getLocation() {
16              return location;
17          }
18
19          public void setLocation(point location) {
20              this.location = location;
21          }
22
23          public int getState() {
24              return state;
25          }
26
27          public void setState(int state) {
28              this.state = state;
29          }
```

Line 15-17: A public method called getLocation that returns a point. That point is the location of the object it is applied to

Line 19-21: A public method called setLocation that takes an input point and assigns it to the location of THIS person it is applied to and returns nothing (void)

Line 23-25: A public method called getState that returns an integer. That integer is the state of the object it is applied to

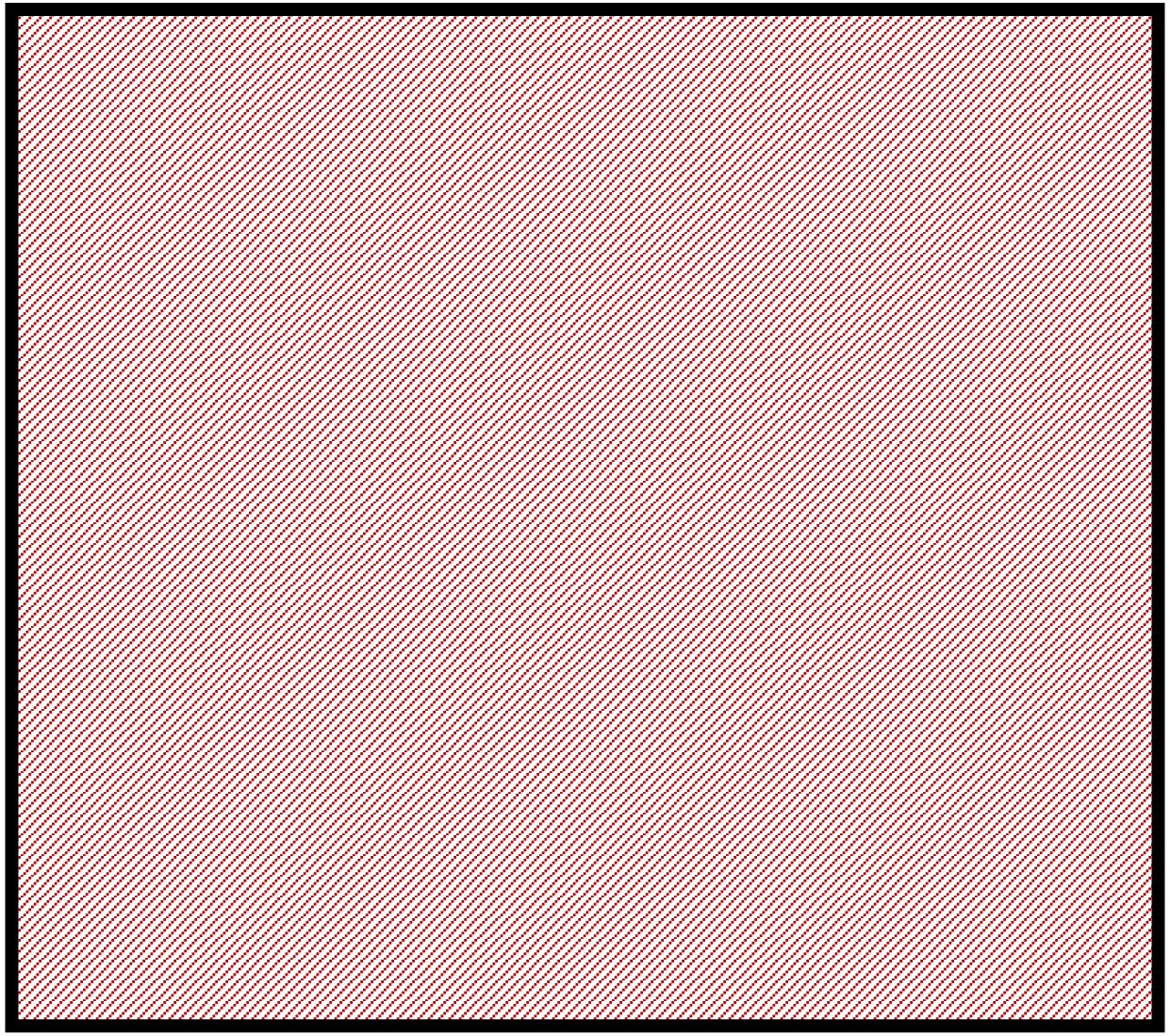
Line 27-29: A public method called setState that takes an input integer and assigns it to the state of THIS person it is applied to and returns nothing (void)

Now lets make a neighborhood!

- What do we know about neighborhoods?
- What is the **minimal yet sufficient** amount of information to describe a neighborhood?

==========

- They have an area
- And they have some number of people (group of many “person”)



Attributes of a Neighborhood

```
3  import java.util.ArrayList;
4
5  public class neighborhood {
6
7      // attributes
8          bbox bb;
9          ArrayList<person> people;
10
```

Line 3: Imports the ArrayList library
Line 5: We create a public class called neighborhood
Line 8: Define an open member variable of type bbox called bb
Line 9: Define an open ArrayList member that can hold objects of type person, we call this ArrayList people.

Constructors, Getters and Setters

```
 5  public class neighborhood {
 6
 7      // attributes
 8          bbox bb;
 9          ArrayList<person> people;
10
11      // Constructors
12          public neighborhood(bbox bb, ArrayList<person> people) {
13              super();
14              this.bb = bb;
15              this.people = people;
16          }
17      // Getters and Setters
18          public bbox getBb() {
19              return bb;
20          }
21          public void setBb(bbox bb) {
22              this.bb = bb;
23          }
24          public ArrayList<person> getPeople() {
25              return people;
26          }
27          public void setPeople(ArrayList<person> people) {
28              this.people = people;
29          }
30
```

Line 12-16: Neighborhood constructor
Line 18-20: Bounding Box getter
Line 21-23: Bounding Box setter
Line 24-26: ‘people’ getter
Line 27-29: ‘people’ setter

Remember: Right Click -> Source to autogenerate

Delegating Methods to the *ArrayList* class

```
30
31      // Delegation
32          public int size() {
33              return people.size();
34          }
35          public person get(int index) {
36              return people.get(index);
37          }
38          public boolean add(person e) {
39              return people.add(e);
40          }
41          public person remove(int index) {
42              return people.remove(index);
43          }
44          public void clear() {
45              people.clear();
46          }
47
```

Line 32-34: When the size method is applied to a neighborhood object, we expect it to return an integer (describing the size). This method applies the ArrayList size method to the ArrayList 'people'.

Line 35-37: When the get method is applied to a neighborhood object, we expect to return the person object at the location of the input integer. This method applies the ArrayList get method to the ArrayList 'people'.

Line 38-40: Apply ArrayList add to people
Line 41-43: Apply ArrayList remove to people
Line 44-46: Apply ArrayList clear to people

In all these cases we are leveraging existing methods. We do not need to write them (or even know how they work!) – this is delegation

Initial testing ….

```
 2
 3 import java.util.ArrayList;
 4
 5 public class test {
 6
 7     public static void main(String[] args) {
 8
 9         bbox bb = new bbox(new point(0,0),
10                            new point(5000,5000));
11
12         neighborhood n = new neighborhood(bb, new ArrayList<person>());
13
14         n.add(new person(new point(500, 500), 2));
15         System.out.println("Size " + n.size());
16         System.out.println("State " + n.get(0).getState());
17         System.out.println("X Location " + n.get(0).getLocation().getX());
18     }
19 }
20
```

```
Problems  @ Javadoc  Declaration  Console
<terminated> test (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 20, 2020, 3:02:16 PM
Size 1
State 2
X Location 500.0
```

Line 3: Import ArrayList
Line 5: Create a public class called test
Line 7: Create a Main method s telling jave where to start execution
Line 9-10: Create a new bounding box called bb from two new points
Line 12: Create a neighborhood using the bb object and a new empty ArrayList of person(s)
Line 14: Use the delegated ‘add’ method to add a new person to the neighborhood located at the new point 500,500 who is infected (state = 2)
Line 15-17: Test some methods and print the, the the console

```
n.add(new person(new point(500, 500), 2));
```

This is super tedious…. Lets think of a better way to allocate the X, Y and infected states using what we know…

==========

- The X and Y should be *random* but *within the bounds* of the bounding box.
- At initiation, any individual person has a 99% chance of not being infected and a 1% chance of being infected.

```
double x = Math.random() * (max - min) + min;
double y = Math.random() * (max - min) + min;
int state = 1;
if(Math.random() > .99) { size = 2; }
```

Remember back to our discussion of random allocation in Java using Math.random(). Using that logic we could generate random X,Y and state values using the following code where max and min are dummy variables (e.g. this code will not work if copied)

Here we chose the exclusive version meaning a person will never be on the Xmax ,Ymax edges

Lets fill in the min and max dummy variables using the getters and setters of the bounding box object (bb)

```
double x = Math.random() * (bb.getXmax() - bb.getXmin()) + bb.getXmin();
double y = Math.random() * (bb.getYmax() - bb.getYmin()) + bb.getYmin();
int state = 1;
if(Math.random() >= .99) { state = 2; }
```

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

        bbox bb = new bbox(new point(0,0), new point(5000,5000));

        neighborhood n = new neighborhood(bb, new ArrayList<person>());

        n.add(new person(new point(500, 500), 2));
        System.out.println("Size " + n.size());
        System.out.println("State " + n.get(0).getState());
        System.out.println("X Location " + n.get(0).getLocation().getX());

        double x = Math.random() * (bb.getXmax() - bb.getXmin()) + bb.getXmin();
        double y = Math.random() * (bb.getYmax() - bb.getYmin()) + bb.getYmin();
        int state = 1;
        if(Math.random() >= .99) { state = 2; }

        n.add(new person(new point(x, y), state));
        System.out.println("Size " + n.size());
        System.out.println("State " + n.get(1).getState());
        System.out.println("X Location " + n.get(1).getLocation().getX());
        System.out.println("Y Location " + n.get(1).getLocation().getY());
    }
}
```

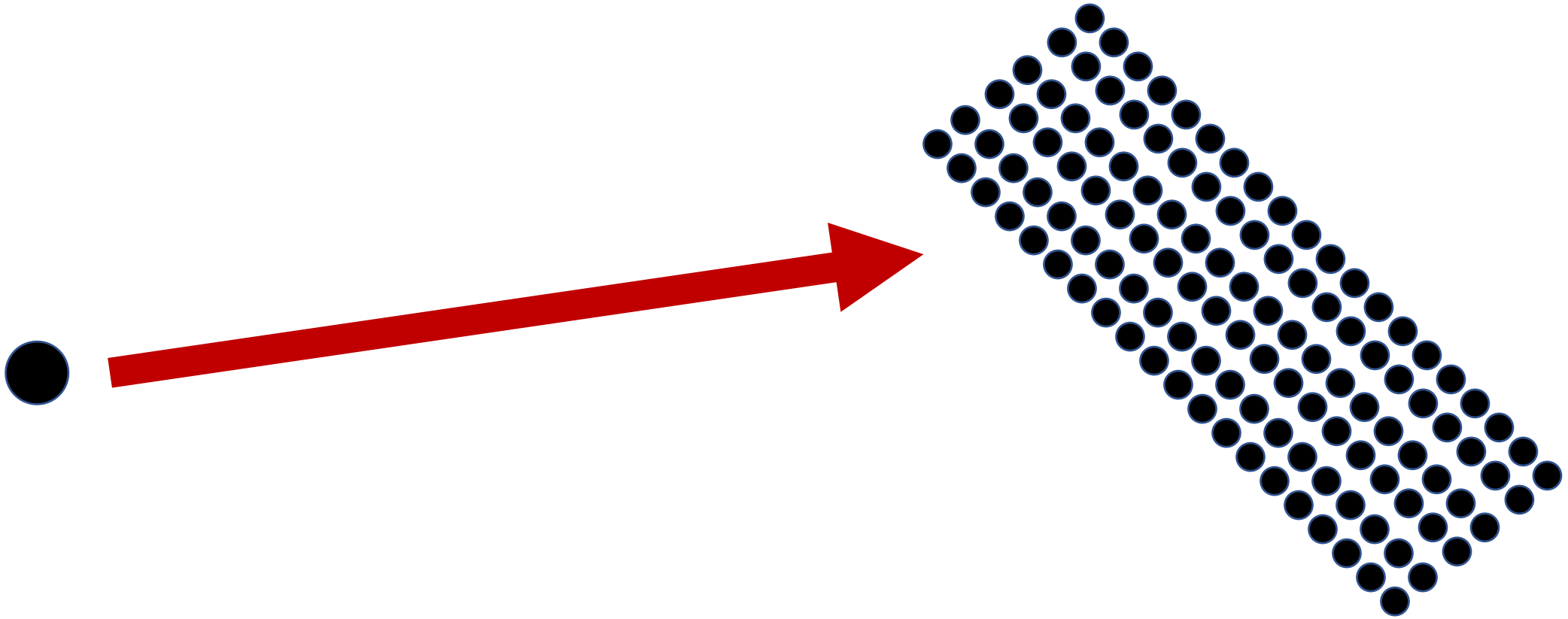
Problems | Javadoc | Declaration | Console

<terminated> test (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java (Apr 20, 2020, 8:52:46 PM – 8:52:47 P

```
Size 1
State 2
X Location 500.0
Size 2
State 1
X Location 3023.1443968388544
Y Location 1487.9342044093669
```

And run some test code!

That’s pretty nice, but lets say we want 500 people in the neighbor hood? How would we deal with that?



Enter the “for-loop”

Here we wrap the randomization and person creation in a loop that runs from 0 > I > 500.

```
4
5  public class test {
6      public static void main(String[] args) {
7
8          bbox bb = new bbox(new point(0,0), new point(5000,5000));
9
10         neighborhood n = new neighborhood(bb, new ArrayList<person>());
11
12         double x, y;
13         int state = 1;
14         int count = 0;
15
16         for (int i = 0; i < 500; i++) {
17             x = Math.random() * (bb.getXmax() - bb.getXmin()) + bb.getXmin();
18             y = Math.random() * (bb.getYmax() - bb.getYmin()) + bb.getYmin();
19             if(Math.random() >= .99) {
20                 state = 2;
21                 count++;
22             }
23
24             n.add(new person(new point(x, y), state));
25         }
26
27
28         System.out.println(n.size() + " people.");
29         System.out.println(count + " infected.");
30
31     }
32 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> test (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 20, 2020, 8:58:03 PM – 8:5
500 people.
6 infected.
```

We then print the size of the neighborhood and count the number of infected people (6/500 = 1.20%).

To those coming from functional programing languages, this looks familiar. However we want the assignment to be part of a object/class rather then in the main method execution…. Why??

What if you wanted to make 10 neighborhoods? Then this becomes quite burdensome….

Sure we could wrap the for-loop in another for-loop but then we are neglecting the power of OOP.

==========

So lets think about what new information we added and what we are doing?

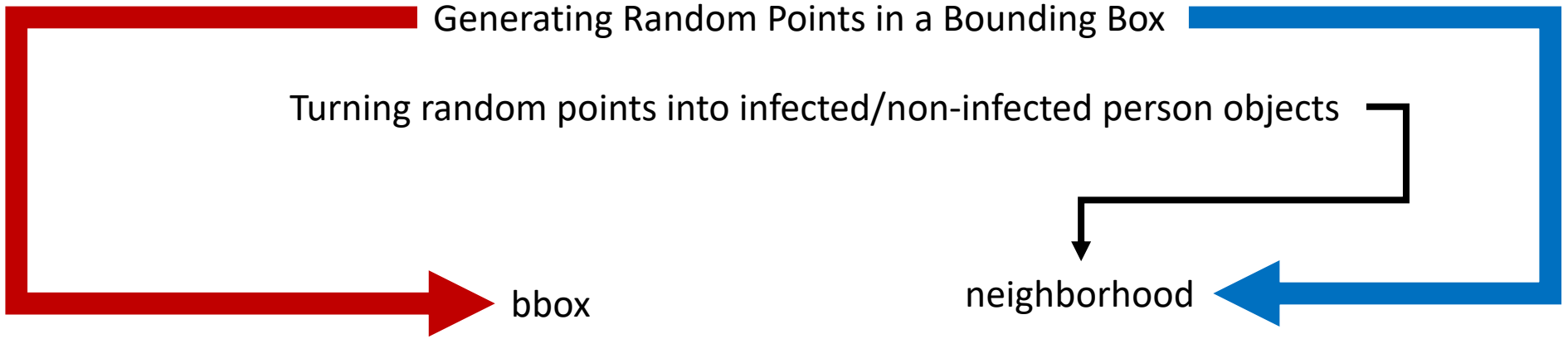
We told Java to add **X** number of people to the neighborhood people array?

We randomly assigned locations to those people as well as an initial infection state based on a global perspective of 1% infection…

People is a member attribute of neighborhood (and so in bounding box!) so all that we needed to add as new information was the number of people that should be and the performed randomization.

That is we created pseudo methods for **addPeople()** and make **randPoint()**

Lets move these over to classes in our model since there are thing we want to repeatedly do….


Generating Random Points in a Bounding Box
Turning random points into infected/non-infected person objects
bbox
neighborhood

My Choice… not the “right” or only one…

```java
public class bbox {

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

    //Constructors
    public bbox(point p1, point p2) {

    public double getXmax() { return xmax; }
    public void   setXmax(double xmax) { this.xmax = xmax; }

    public double getXmin() { return xmin; }
    public void   setXmin(double xmin) { this.xmin = xmin; }

    public double getYmax() { return ymax; }
    public void   setYmax(double ymax) { this.ymax = ymax; }

    public double getYmin() { return ymin; }
    public void   setYmin(double ymin) { this.ymin = ymin; }


    public boolean isInside(point p) {

    public point randPoint() {
      double x = Math.random() * ((this.getXmax() - this.getXmin()) + 1) + this.getXmin()
      double y = Math.random() * ((this.getYmax() - this.getYmin()) + 1) + this.getYmin()
      return new point(x,y);
    }
}
```

```java
public class neighborhood {
    // attributes
        bbox bb;
        ArrayList<person> people;

    // Constructors
        public neighborhood(bbox bb, ArrayList<person> people) {
            this.bb = bb;
            this.people = people;
        }
    // Getters and Setters
        public bbox getBb() { return bb; }
        public void setBb(bbox bb) { this.bb = bb; }
        public ArrayList<person> getPeople() { return people; }
        public void setPeople(ArrayList<person> people) { this.people = people; }

    // Delegation
        public int size() { return people.size(); }
        public person get(int index) { return people.get(index); }
        public boolean add(person e) { return people.add(e); }
        public person remove(int index) { return people.remove(index);}
        public void clear() {people.clear();}

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

Why?

- ***randPoint*** is not a neighborhood specific method but a bounding box specific method. It is something that would be useful in other models (say modeling invasive species, or really any Agent-based modeling exercise)
- *This is looking ahead to ideas of inheritance!*
- *FYI NetLogo calls this idea "Breeding turtles" where turtles are agents.*

```
4
5  public class test {
6
7      public static void main(String[] args) {
8
9          bbox bb = new bbox(new point(0,0), new point(5000,5000));
10
11         neighborhood n = new neighborhood(bb, new ArrayList<person>());
12
13         n.addPeople(500);
14             System.out.println(n.size());
15         n.addPeople(500);
16             System.out.println(n.size());
17         n.clear();
18             System.out.println(n.size());
19         n.addPeople(500);
20             System.out.println(n.size());
21         //System.out.println(count);
22     }
23 }
24
```

Problems | Javadoc | Declaration | Console

```
<terminated> test (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java  (Apr 20, 2020, 3:35:40
500
1000
0
500
```

And run some test code!

Homework Hints

1. Create a method that determines if a person is within X distance of another person.

public boolean isWithin(point p1, double distance) {…}

- Is *isWithin* a person or a point method?

2. Create a method that counts the number of infected people in a neighborhood at any one time:

public int numInfected(){ …}

Homework Hints

3. In your test class, write code that checks if person(s) are within 6 feet of infected person(s), and, if so, change their state to infected.
    - You will need to use your new is*Within* method, getters, setters, and delegations

- This only needs to be checked once! You do not need to move people or run the check multiple times. When complete you should be able to print a message like the following that changes each time its run (thanks to successful randomization)

```
There are 500 people in the neighborhod
4 are infected
5 new infections
====================
9 total
```

```
There are 500 people in the neighborhod
5 are infected
6 new infections
====================
11 total
```