```
class Point { ... }
interface Region { public boolean contains(Point p); }
class RectRegion implements Region {
  Point lowerLeft, upperRight;
  RectRegion(Point lowerL, Point upperR) { ... }
  public boolean contains(Point p) { ... }
}
class CircleRegion implements Region {
  Point center; int radius;
  CircleRegion(Point center, int radius) { ... }
  public boolean contains(Point p) { ... }
}
class RegionExamples {
  // Takes a region and an array of points and returns a new array of the points that are within that region.
  static Point[] pointsWithin(Region r, Point[] p) {









  }
  Region circ = new CircleRegion(new Point(40, 40), 20);
  Region rect = new RectRegion(new Point(10, 10), new Point(50, 50));

  Point[] examplePoints = {new Point(35, 45), new Point(100, 100), new Point(51, 51)};

  Point[] withinCirc = RegionExamples.pointsWithin(this.circ, this.examplePoints);
  Point[] withinRect = RegionExamples.pointsWithin(this.rect, this.examplePoints);
}
```

1. What are the expected values of `withinCirc` and `withinRect`?

2. Implement `pointsWithin`

3. Draw a memory diagram of the heap after the examples have run.

4. Draw a trace of the call to `pointsWithin` on `rect`.

Name: _____ PID: _____ Code: _____

```
... code from front ...
```

Challenge: Implement a main class for the command-line behavior below.

```
class RegionMain {









  public static void main(String[] args) {

















  }
}
```

```
$ javac Region.java
$ java RegionMain circle 40 40 20 35 45 100 100 51 51
Inside the region: 35, 45; 51, 51
$ java RegionMain rectangle 10 10 50 50 35 45 100 100 51 51
Inside the region: 35, 45
```