```
class Point {
  private double x, y;
  Point(double x, double y) { this.x = x; this.y = y; }
  double distance(Point other) {
    return Math.sqrt(
      Math.pow(this.x - other.x, 2) +
      Math.pow(this.y - other.y, 2));
  }
  double xDistance(Point other) { return Math.abs(other.x - this.x); }
  double yDistance(Point other) { return Math.abs(other.y - this.y); }
}
interface Region {
  boolean contains(Point p);
  Region add(Region other);
  Region overlap(Region other);
}
abstract class ARegion implements Region {
  public Region add(Region other) {
    return new UnionRegion(this, other);
  }
  public Region overlap(Region other) {
    return new IntersectRegion(this, other);
  }
}
class UnionRegion extends ARegion {
  Region region1, region2;
  UnionRegion(Region region1, Region region2) {
    this.region1 = region1;
    this.region2 = region2;
  }
  public boolean contains(Point toCheck) {
    return this.region1.contains(toCheck) ||
           this.region2.contains(toCheck);
  }
}
class IntersectRegion extends ARegion {
  Region region1, region2;
  IntersectRegion(Region region1, Region region2) {
    this.region1 = region1;
    this.region2 = region2;
  }
  public boolean contains(Point toCheck) {
    return this.region1.contains(toCheck) &&
           this.region2.contains(toCheck);
  }
}
class SubtractRegion implements Region {
  Region region1;
  Region region2;
  SubtractRegion(Region region1, Region region2) {
    this.region1 = region1;
    this.region2 = region2;
  }
  public boolean contains(Point toCheck) {
    return this.region1.contains(toCheck) &&
           !this.region2.contains(toCheck);
  }
}
class SquareRegion extends ARegion {
  Point center;
  double sideLength;
  SquareRegion(Point center, double sideLength) {
    this.center = center;
    this.sideLength = sideLength;
  }
  public boolean contains(Point toCheck) {
    return this.center.xDistance(toCheck) <= (this.sideLength / 2) &&
           this.center.yDistance(toCheck) <= (this.sideLength / 2);
  }
}
class CircleRegion extends ARegion {
  Point center;
  double radius;
  CircleRegion(Point center, double radius) {
    this.center = center;
    this.radius = radius;
  }
  public boolean contains(Point toCheck) {
    return this.center.distance(toCheck) <= this.radius;
  }
}
class ExamplesARegion {
  Region circ1 = new CircleRegion(new Point(10, 5), 4.0);
  Region sq = new SquareRegion(new Point(10, 1), 8.0);
  Region ir = this.circ1.add(this.sq);
}
```

```
class UnionRegion extends ARegion {
  Region r1, r2;
  UnionRegion(Region r1, Region r2) {
    this.r1 = r1;
    this.r2 = r2;
  }
  public boolean contains(Point p) {
     return this.r1.contains(p) &&
            this.r2.contains(p);
  }
}
```

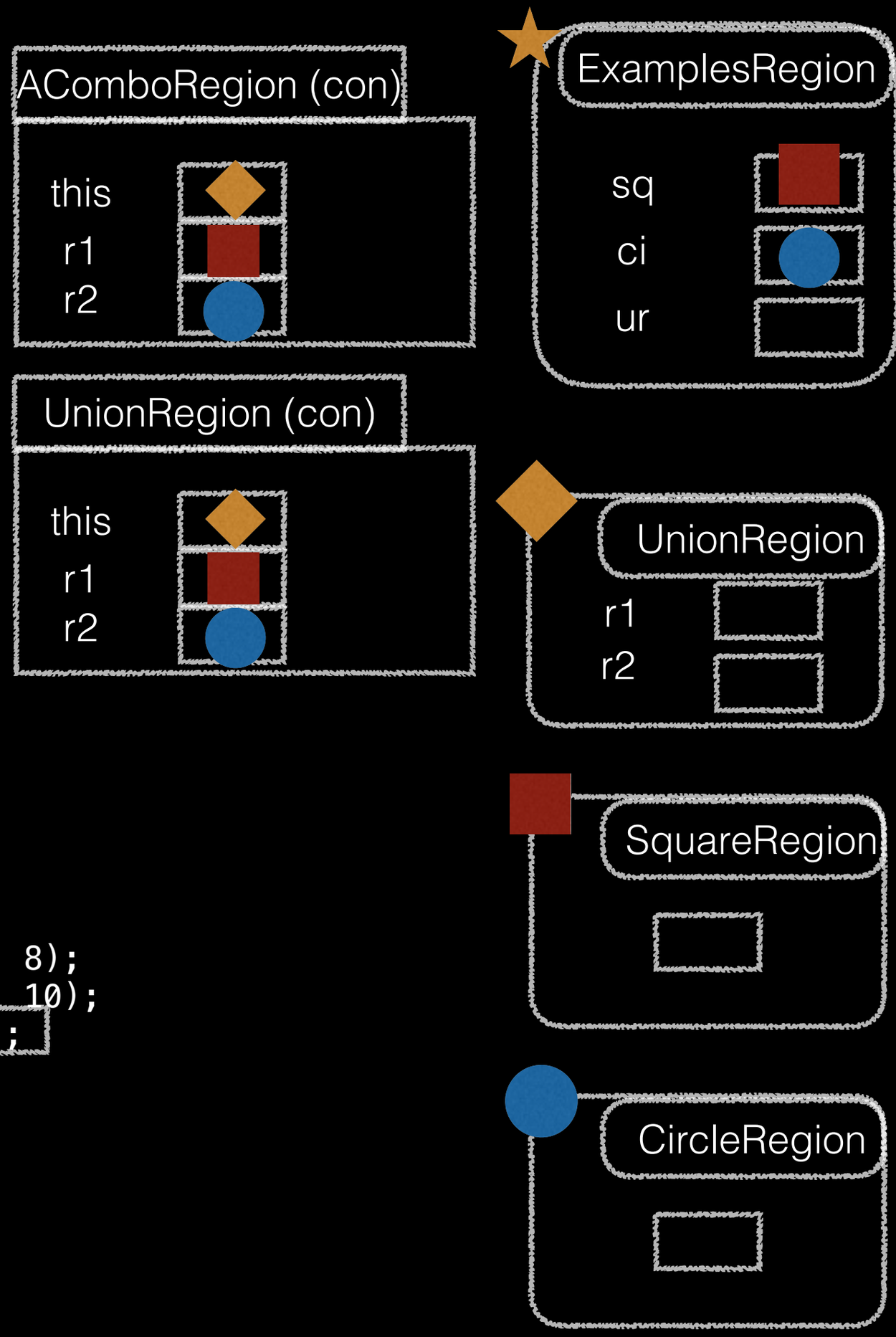
```
class IntersectRection extends ARegion {
  Region r1, r2;
  IntersectRegion(Region r1, Region r2) {
    this.r1 = r1;
    this.r2 = r2;
  }
  public boolean contains(Point p) {
     return this.r1.contains(p) ||
            this.r2.contains(p);
  }
}
```

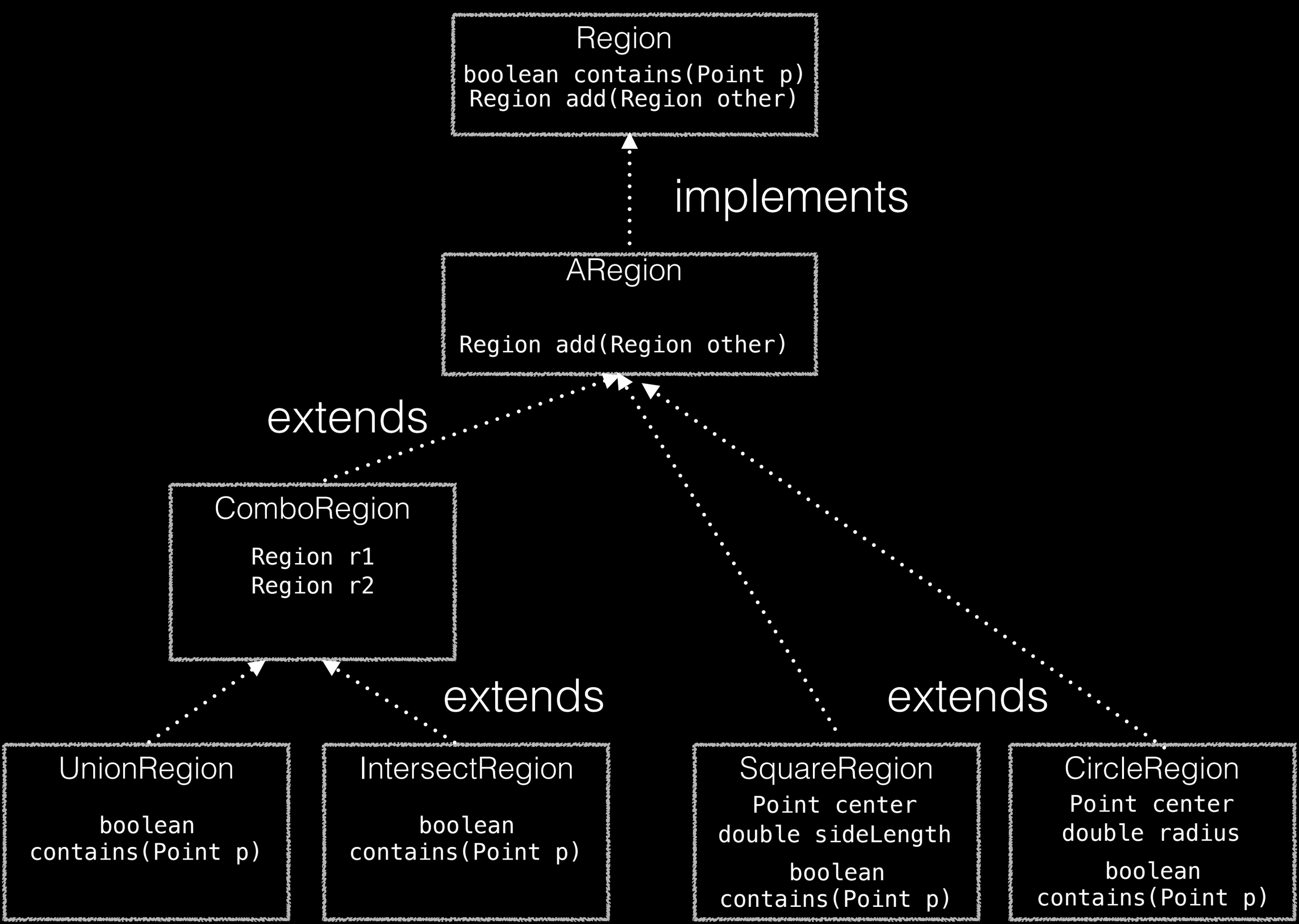
```
class UnionRegion extends AComboRegion {

  UnionRegion(Region r1, Region r2) {
    super(r1, r2);
  }
  public boolean contains(Point p) {
     return this.r1.contains(p) &&
            this.r2.contains(p);
  }
}
```

```
abstract class AComboRegion {
  Region r1, r2;
  AComboRegion(Region r1, Region r2) {
    this.r1 = r1;
    this.r2 = r2;
  }
}
```

```
class IntersectRection extends AComboRegion {

  IntersectRegion(Region r1, Region r2) {
    super(r1, r2);
  }
  public boolean contains(Point p) {
     return this.r1.contains(p) ||
            this.r2.contains(p);
  }
}
```

```
interface Region { … }
abstract class ARegion implements Region { … }
class SquareRegion extends ARegion { … }
class CircleRegion extends ARegion { … }

abstract class AComboRegion extends ARegion {
  Region r1, r2;
  AComboRegion(Region r1, Region r2) {
    this.r1 = r1;
    this.r2 = r2;
  }
}

class UnionRegion extends AComboRegion {
  UnionRegion(Region r1, Region r2) {
    super(r1, r2);
  }
  public boolean contains(Point p) {
     return this.r1.contains(p) &&
            this.r2.contains(p);
  }
}

class ExamplesRegion {
  Region sq = new SquareRegion(new Point(4, 5), 8);
  Region ci = new CircleRegion(new Point(6, 7), 10);
  Region ur = new UnionRegion(this.sq, this.ci);
}
```




Region
boolean contains(Point p)
Region add(Region other)
implements
ARegion
Region add(Region other)
extends
extends
extends
ComboRegion
Region r1
Region r2
UnionRegion
boolean
contains(Point p)
IntersectRegion
boolean
contains(Point p)
SquareRegion
Point center
double sideLength
boolean
contains(Point p)
CircleRegion
Point center
double radius
boolean
contains(Point p)

Region

```
boolean contains(Point p)
Region add(Region other)
```

implements

ARegion

```
Region add(Region other)
```

extends

extends

ComboRegion

```
Region r1
Region r2
```

extends

ShapeRegion

```
Point center
```

extends

UnionRegion

```
boolean
contains(Point p)
```

IntersectRegion

```
boolean
contains(Point p)
```

SquareRegion

```
double sideLength
boolean
contains(Point p)
```

CircleRegion

```
double sideLength
boolean
contains(Point p)
```