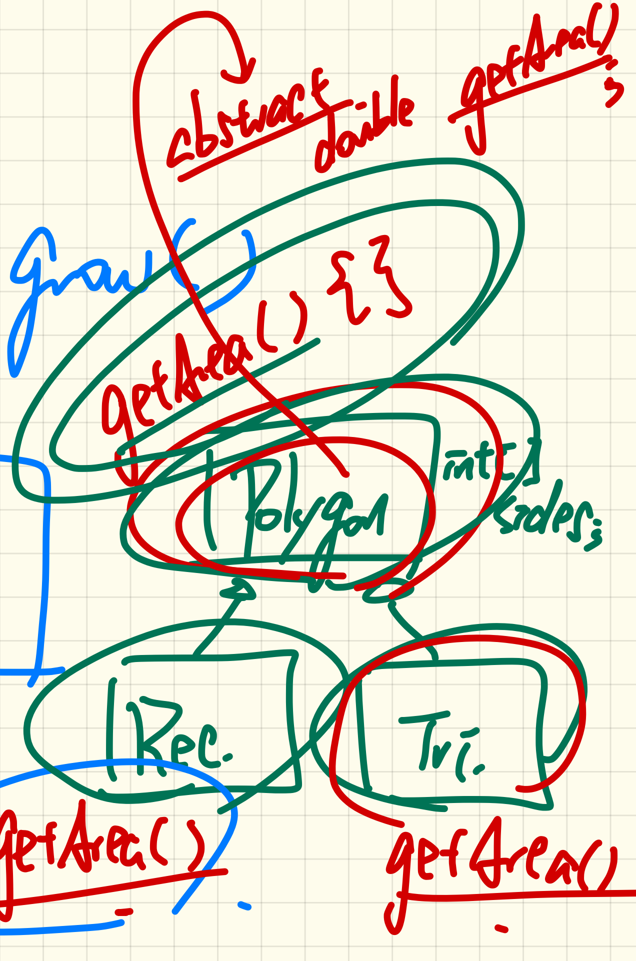
LECTURE 21

FRIDAY NOVEMBER 22

Polygon p = new Rectangle();

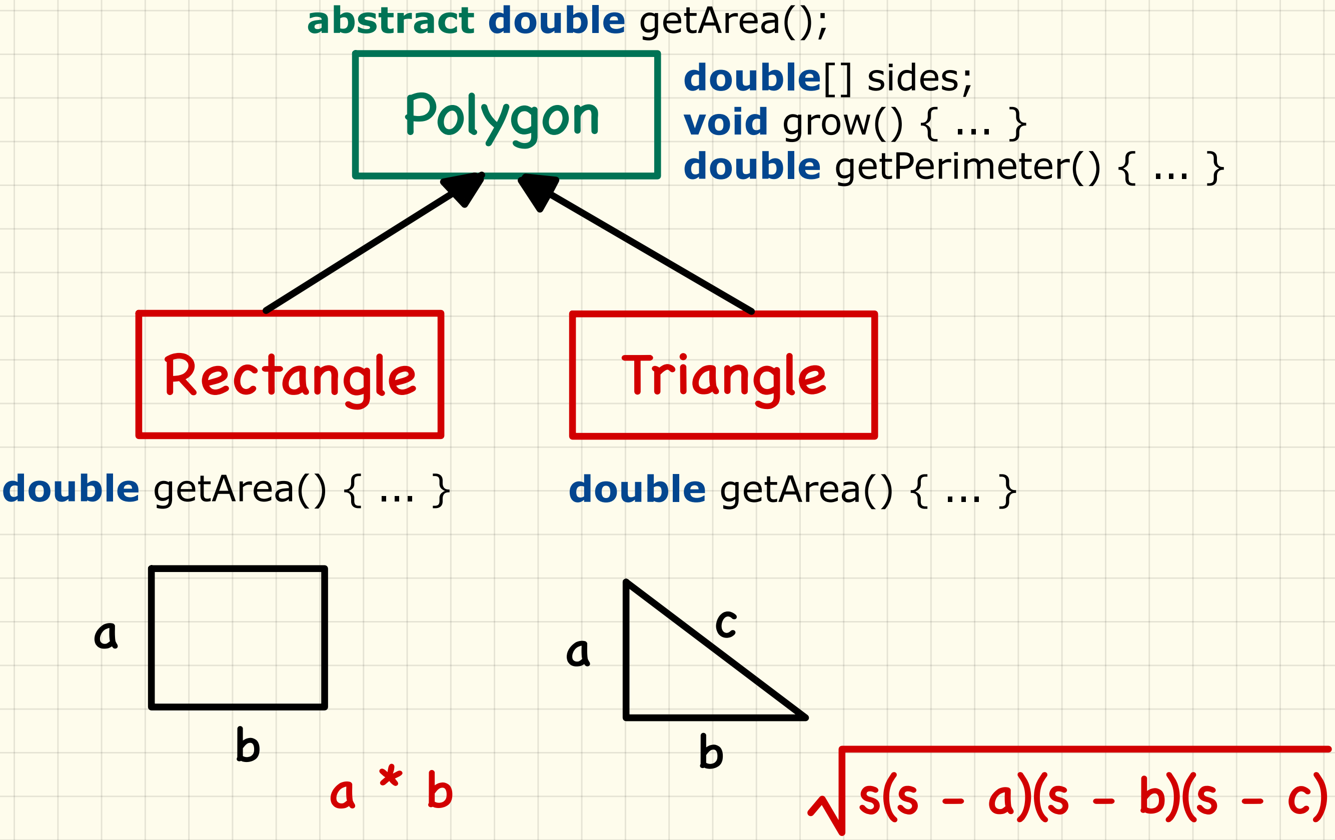
println(p.getArea());

ST: Polygon



getArea
getPerimeter.

Abstract Implementation vs. Concrete Implementation



abstract double getArea();

Polygon

double[] sides;
void grow() { … }
double getPerimeter() { … }

Rectangle

Triangle

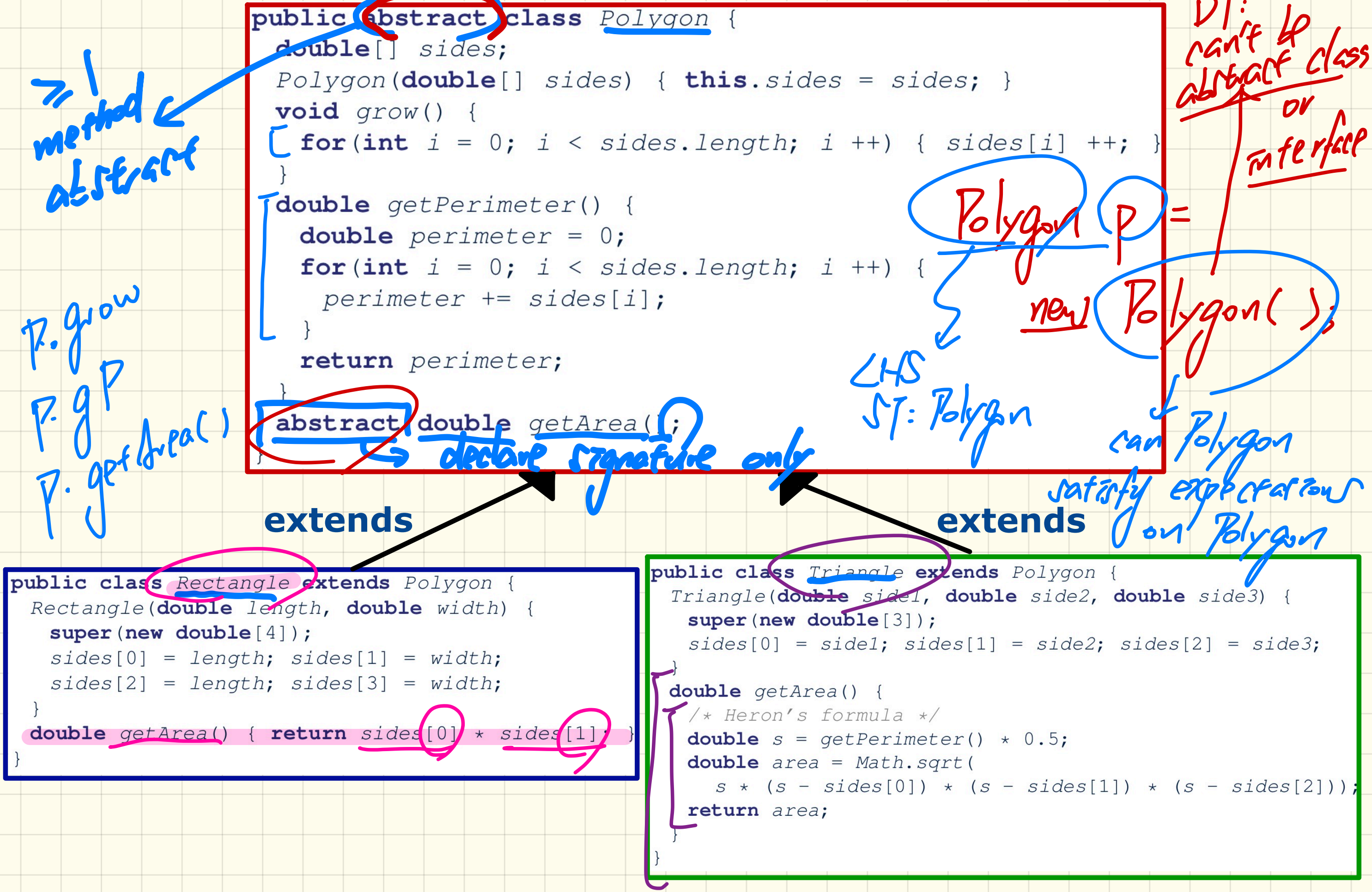
double getArea() { … }

double getArea() { … }

a * b

$\sqrt{s(s - a)(s - b)(s - c)}$

Abstract Class vs. Concrete Descendants



```
public abstract class Polygon {
  double[] sides;
  Polygon(double[] sides) { this.sides = sides; }
  void grow() {
   for(int i = 0; i < sides.length; i ++) { sides[i] ++; }
  }
  double getPerimeter() {
   double perimeter = 0;
   for(int i = 0; i < sides.length; i ++) {
    perimeter += sides[i];
   }
   return perimeter;
  }
  abstract double getArea();
}
```

extends

```
public class Rectangle extends Polygon {
 Rectangle(double length, double width) {
  super(new double[4]);
  sides[0] = length; sides[1] = width;
  sides[2] = length; sides[3] = width;
 }
 double getArea() { return sides[0] * sides[1]; }
}
```

extends

```
public class Triangle extends Polygon {
 Triangle(double side1, double side2, double side3) {
  super(new double[3]);
  sides[0] = side1; sides[1] = side2; sides[2] = side3;
 }
 double getArea() {
  /* Heron's formula */
  double s = getPerimeter() * 0.5;
  double area = Math.sqrt(
    s * (s - sides[0]) * (s - sides[1]) * (s - sides[2]));
  return area;
 }
}
```

≥ 1 method abstract
declare signature only
P.grow
P.gP
P.getArea()
DT: can't be abstract class or interface
Polygon p = new Polygon( );
LHS
ST: Polygon
can Polygon satisfy expectations on Polygon

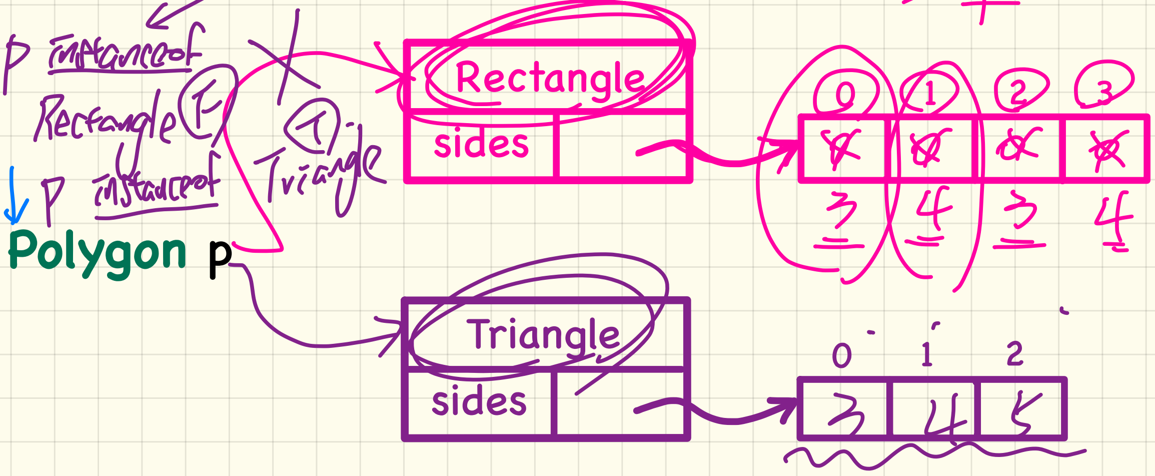
Polymorphic Assignments of Polygons

```
public abstract class Polygon {
  double[] sides;
  Polygon(double[] sides) { this.sides = sides; }
  void grow() {
    for(int i = 0; i < sides.length; i ++) { sides[i] ++; }
  }
  double getPerimeter() {
    double perimeter = 0;
    for(int i = 0; i < sides.length; i ++) {
      perimeter += sides[i];
    }
    return perimeter;
  }
  abstract double getArea();
}
```

Polygon (new double[4]);
sides[0] = 3;
sides[1] = 4;

```
Polygon p;
p = new Rectangle(3, 4); /* polymorphism */
System.out.println(p.getPerimeter()); /* 14.0 */
System.out.println(p.getArea()); /* 12.0 */
p = new Triangle(3, 4, 5); /* polymorphism */
System.out.println(p.getPerimeter()); /* 12.0 */
System.out.println(p.getArea()); /* 6.0 */
```

p instanceof Rectangle T.

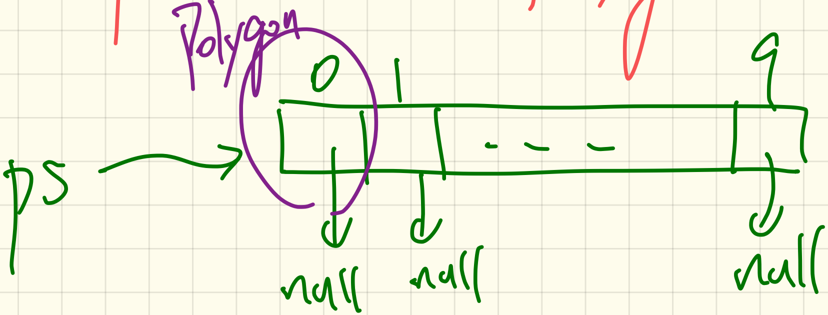


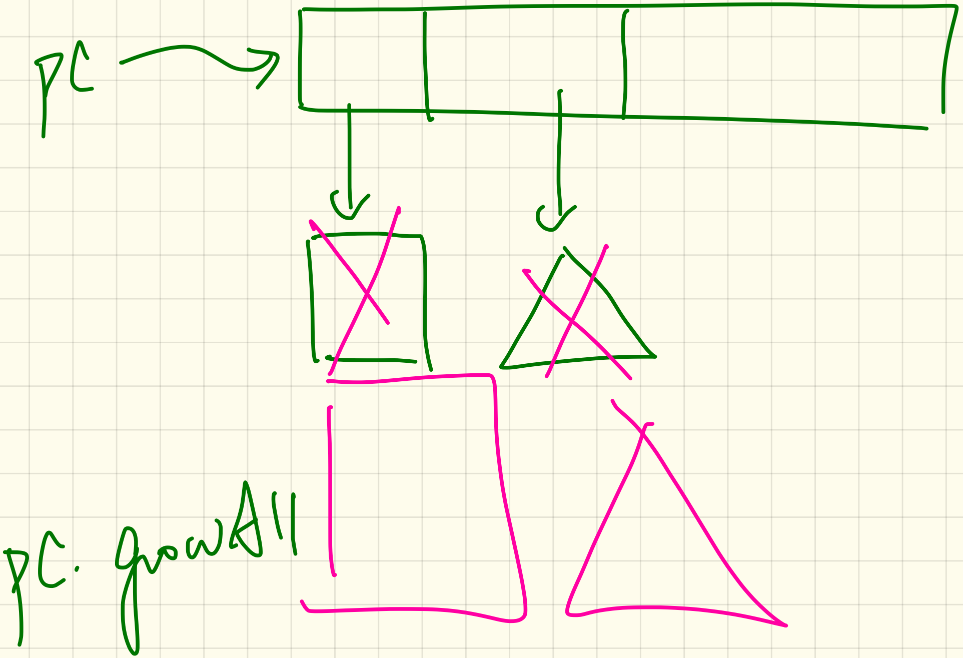
Polygon p = new Polygon(); X

abstract class

✓.

Polygon[] ps = new Polygon[10];





pc.growAll

Polymorphic Collection of Polygons

col.polygons[1] instanceof Polygon T

col.polygons[1] instanceof Rectangle F

col.polygons[1] instanceof Triangle T

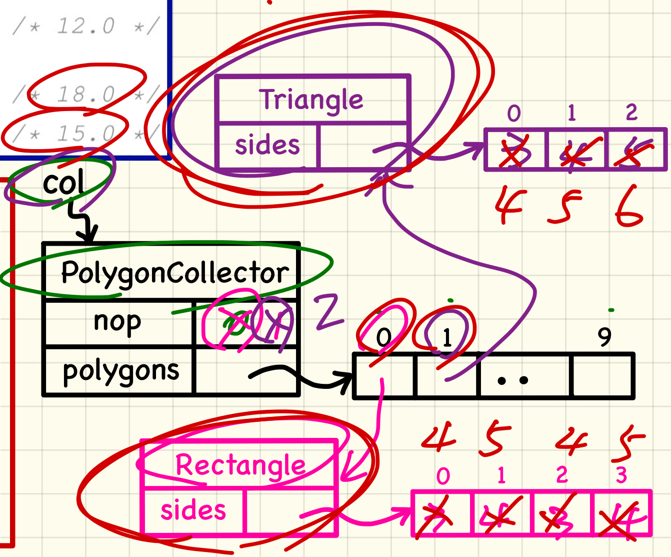
```java
public abstract class Polygon {
  double[] sides;
  Polygon(double[] sides) { this.sides = sides; }
  void grow() {
    for(int i = 0; i < sides.length; i ++) { sides[i] ++; }
  }
  double getPerimeter() {
    double perimeter = 0;
    for(int i = 0; i < sides.length; i ++) {
      perimeter += sides[i];
    }
    return perimeter;
  }
  abstract double getArea();
}
```

```java
PolygonCollector col = new PolygonCollector();
col.addPolygon(new Rectangle(3, 4)); /* polymorphism */
col.addPolygon(new Triangle(3, 4, 5)); /* polymorphism */
System.out.println(col.polygons[0].getPerimeter()); /* 14.0 */
System.out.println(col.polygons[1].getPerimeter()); /* 12.0 */
col.growAll();
System.out.println(col.polygons[0].getPerimeter()); /* 18.0 */
System.out.println(col.polygons[1].getPerimeter()); /* 15.0 */
```

```java
public class PolygonCollector {
  Polygon[] polygons;
  int numberOfPolygons;
  PolygonCollector() { polygons = new Polygon[10]; }
  void addPolygon(Polygon p) {
    polygons[numberOfPolygons] = p; numberOfPolygons ++;
  }
  void growAll() {
    for(int i = 0; i < numberOfPolygons; i ++) {
      polygons[i].grow();
    }
  }
}
```

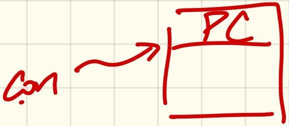
0

1



Polymorphic Return Value of Polygons

Polygon p;



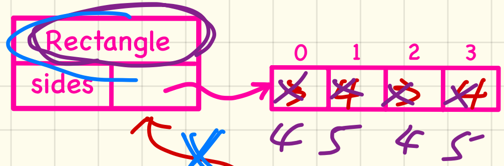
```
PolygonConstructor con = new PolygonConstructor();
double[] recSides = {3, 4, 3, 4}; p = con.getPolygon(recSides);
System.out.println(p instanceof Polygon);
System.out.println(p instanceof Rectangle);
System.out.println(p instanceof Triangle);
System.out.println(p.getPerimeter()); /* 14.0 */
System.out.println(p.getArea()); /* 12.0 */
con.grow(p);
System.out.println(p.getPerimeter()); /* 18.0 */
System.out.println(p.getArea()); /* 20.0 */
double[] triSides = {3, 4, 5}; p = con.getPolygon(triSides);
System.out.println(p instanceof Polygon);
System.out.println(p instanceof Rectangle);
System.out.println(p instanceof Triangle);
System.out.println(p.getPerimeter()); /* 12.0 */
System.out.println(p.getArea()); /* 6.0 */
con.grow(p);
System.out.println(p.getPerimeter()); /* 15.0 */
System.out.println(p.getArea()); /* 9.921 */
```

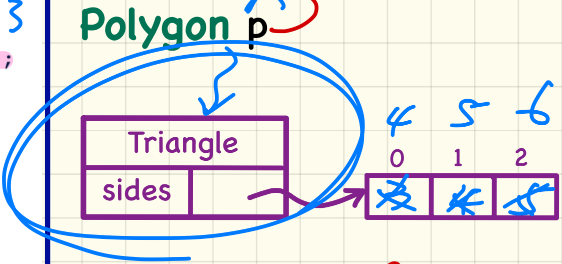
↳ ST: Polygon

✓ ✓ ✗

{3,4,5}

✓ ✗ ✓





```
public abstract class Polygon {
  double[] sides;
  Polygon(double[] sides) { this.sides = sides; }
  void grow() {
    for(int i = 0; i < sides.length; i ++) { sides[i] ++; }
  }
  double getPerimeter() {
    double perimeter = 0;
    for(int i = 0; i < sides.length; i ++) {
      perimeter += sides[i];
    }
    return perimeter;
  }
  abstract double getArea();
}
```

```
public class PolygonConstructor {
  Polygon getPolygon(double[] sides) {
    Polygon p = null;
    if (sides.length == 3) {
      p = new Triangle(sides[0], sides[1], sides[2]);
    }
    else if(sides.length == 4) {
      p = new Rectangle(sides[0], sides[1]);
    }
    return p;
  }
  void grow(Polygon p) { p.grow(); }
}
```

{3,4,3,4}

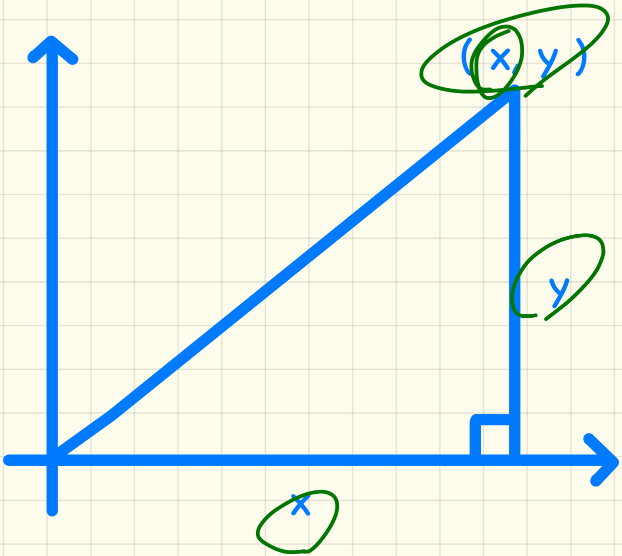
{3,4,5}

3 4 5

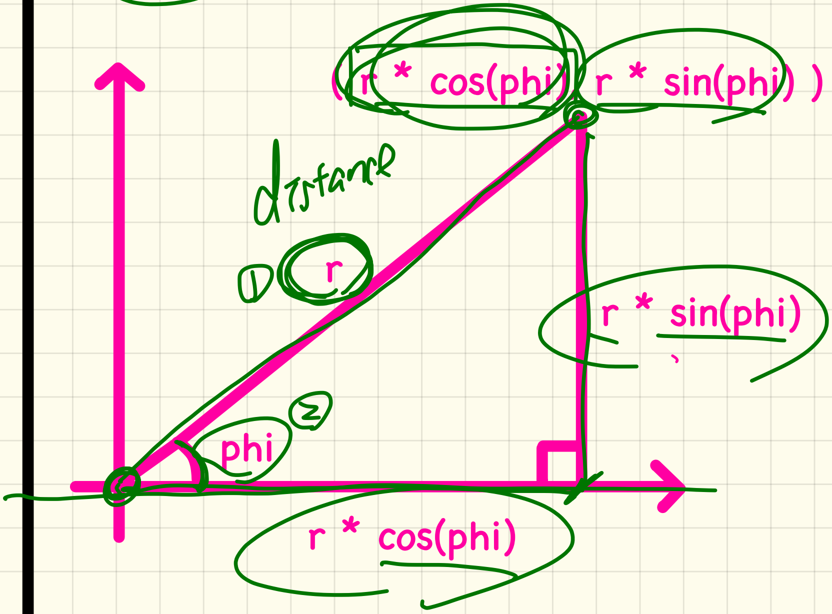
3 4

Representations of 2-D Points: Cartesian vs. Polar

Cartesian System



Polar System

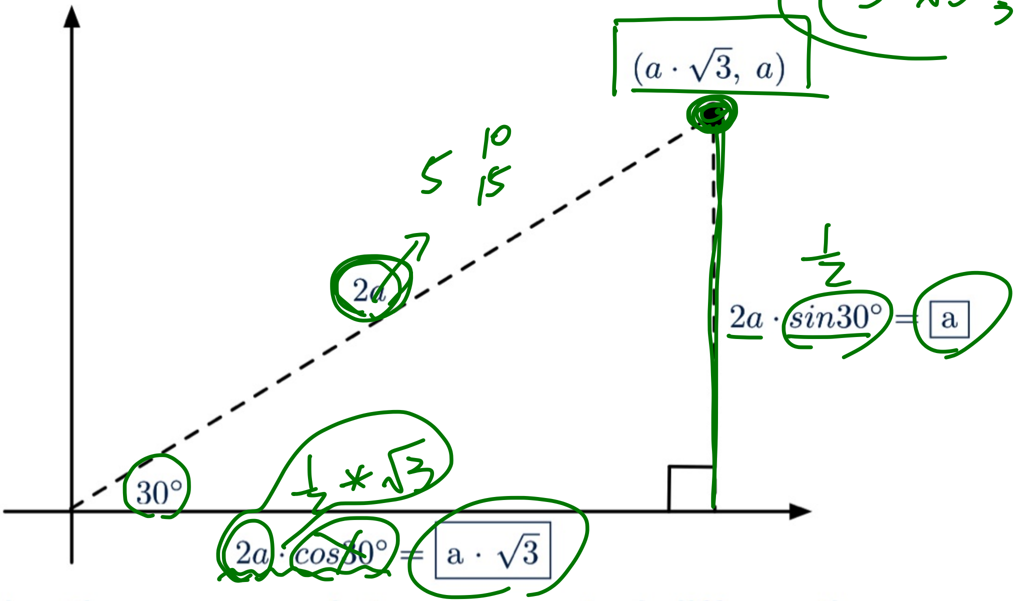


Cartesian vs. Polar: Example

a = 5.

Recall: $sin30^\circ = \frac{1}{2}$ and $cos30^\circ = \frac{1}{2} \cdot \sqrt{3}$

$(5 \cdot \sqrt{3}, 5)$



$(a \cdot \sqrt{3},\ a)$
5
10
15
2a
$\frac{1}{2}$
$2a \cdot sin30^\circ = \boxed{a}$
30°
$\frac{1}{2} * \sqrt{3}$
$2a \cdot cos30^\circ = \boxed{a \cdot \sqrt{3}}$

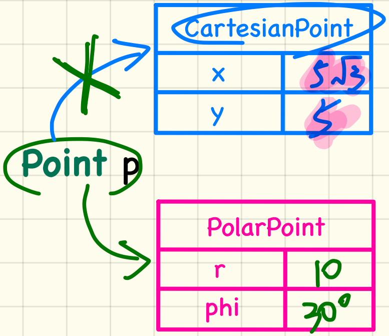
We consider the same point represented differently as:

- $r = 2a$, $\psi = 30^\circ$ [polar system]
- $x = 2a \cdot cos30^\circ = a \cdot \sqrt{3}$, $y = 2a \cdot sin30^\circ = a$ [cartesian system]

```
Point p = new ~~Point~~()
```

interface.

```
      = ✓ new CartesianPoint()
        ✓ new PolarPoint()
```



CartesianPoint	
x	5√3
y	5

PolarPoint	
r	10
phi	30°

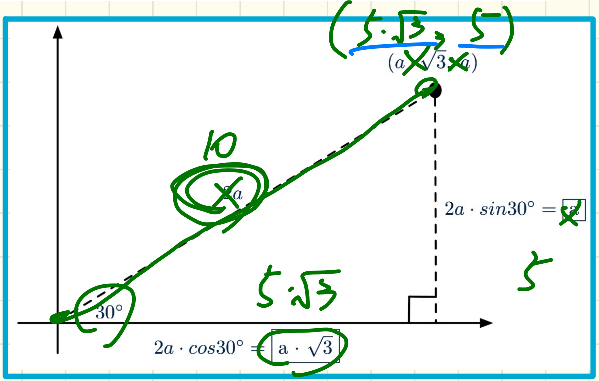
```
interface Point {
        double getX();
        double getY();
}
```

implements   implements

```
public class CartesianPoint implements Point {
  double x;
  double y;
  CartesianPoint(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public double getX() { return x; }
  public double getY() { return y; }
}
```

```
public class PolarPoint implements Point {
  double phi;
  double r;
  public PolarPoint(double r, double phi) {
    this.r = r;
    this.phi = phi;
  }
  public double getX() { return Math.cos(phi) * r; }
  public double getY() { return Math.sin(phi) * r; }
}
```

```
double A = 5;
double X = ~~A * Math.sqrt(3)~~; 5·√3
double Y = ~~A~~; 5
Point p;
                             5√3  5
p = new CartisianPoint(X, Y); /* polymorphism */
print("(" + p.getX() + ", " + p.getY() + ")");
           10
p = new PolarPoint(2 * A, Math.toRadians(30)); /
print("(" + p.getX() + ", " + p.getY() + ")");
```


(5·√3, 5)
(a√3, a)
10
2a
2a · sin30° = a
5
5·√3
30°
2a · cos30° = a · √3