CENG 1004
Introduction to Object Oriented Programming

Spring 2018

Özgür Kılıç
Office: E1-03
Office hours: Mon 13:30-15:20
email: ozgur.kilic10@gmail.com

Course Web Page:
piazza.com/mu.edu.tr/spring2018/ceng1004/home

Sign Up Link
piazza.com/mu.edu.tr/spring2018/ceng1004

Defining Classes

Let’s declare a Point Class

```
public class Point{




}
```

Note

- Class names are Capitalized
- 1 Class = 1 file
- Having a main method means the class can be run

Let's declare a Point Class

```
public class Point{

}
```

Let's declare a Point Class

```
public class Point {

    TYPE var_name;
    TYPE var_name = some_value;



}
```

Let's declare a Point Class

```
public class Point {

    int xCoordinate;
    int yCoordinate;

}
```

Class Definition

Ok, let's create a point instance!

```
Point point1 = new Point();
```

Class

Instance

Ok, let's create a point instance!

```
Point point1 = new Point();
```

What about the coordinates of the point?

Constructors

```
public class CLASSNAME{
    CLASSNAME ( ) {
    }

    CLASSNAME ([ARGUMENTS]) {
    }
}
```

```
CLASSNAME obj1 = new CLASSNAME();
CLASSNAME obj2 = new CLASSNAME([ARGUMENTS])
```

Constructors

- Constructor name == the class name
- No return type – never returns anything
- Usually initialize fields
- All classes need at least one constructor
  - If you don't write one, defaults to

```
CLASSNAME () {
}
```

Point Constructor

```
public class Point {

    int xCoordinate;
    int yCoordinate;

    public Point(int x, int y){
        xCoordinate = x;
        yCoordinate = y;
    }


}
```

Point methods

```
public class Point {

    int xCoordinate;
    int yCoordinate;

    public Point(int x, int y){
        xCoordinate = x;
        yCoordinate = y;
    }

    public void move(int xDistance, int yDistance){
        xCoordinate += xDistance;
        yCoordinate += yDistance;
    }
}
```

Point Class

```
public class Point {

    int xCoordinate;
    int yCoordinate;

    public Point(int x, int y){
        xCoordinate = x;
        yCoordinate = y;
    }

    public void move(int xDistance, int yDistance){...}
    public double distanceFromOrigin(){...}
    public double distanceFromPoint(Point point){...}
    ...
}
```

Class Definition

Using Classes

Classes and Instances

```
public class Test {

    public static void main(String[] args) {

        Point point1 = new Point(10,10);
        Point point2 = new Point(15, 22);

    }

}
```

Accessing fields

- Object.FIELDNAME

```
public class Test {

    public static void main(String[] args) {
        Point point1 = new Point(10,10);
        Point point2 = new Point(15, 22);

        System.out.println("x: " + point1.xCoordinate + ", y: " + point1.yCoordinate);

    }

}
```

Calling Methods

- Object.METHODNAME([ARGUMENTS])

```
public static void main(String[] args) {
    Point point1 = new Point(10,10);
    Point point2 = new Point(15, 22);

    point1.move(5, 5);

    System.out.println("x: " + point1.xCoordinate + ", y: " + point1.yCoordinate);

}
```

References vs Values

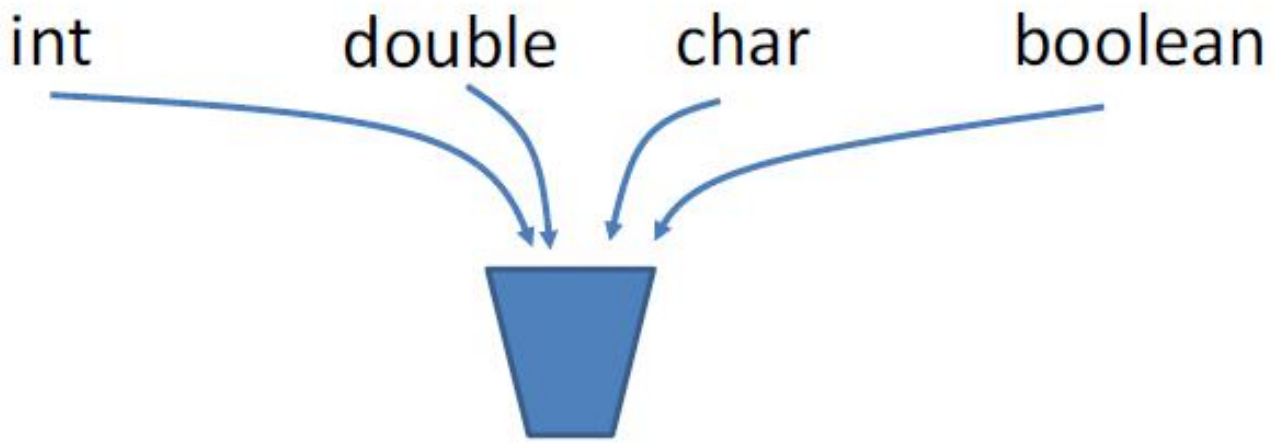
Primitives vs References

- Primitive types are basic java types
  - int, long, double, boolean, char, short, byte, float
  - The actual values are stored in the variable

- Reference types are arrays and objects
  - String, int[], Baby, …

How java stores primitives

- Variables are like fixed size cups
- Primitives are small enough that they just fit into the cup

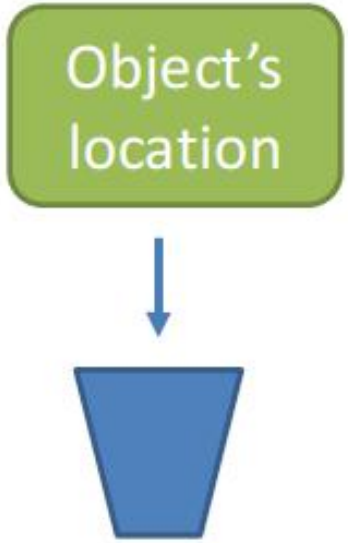


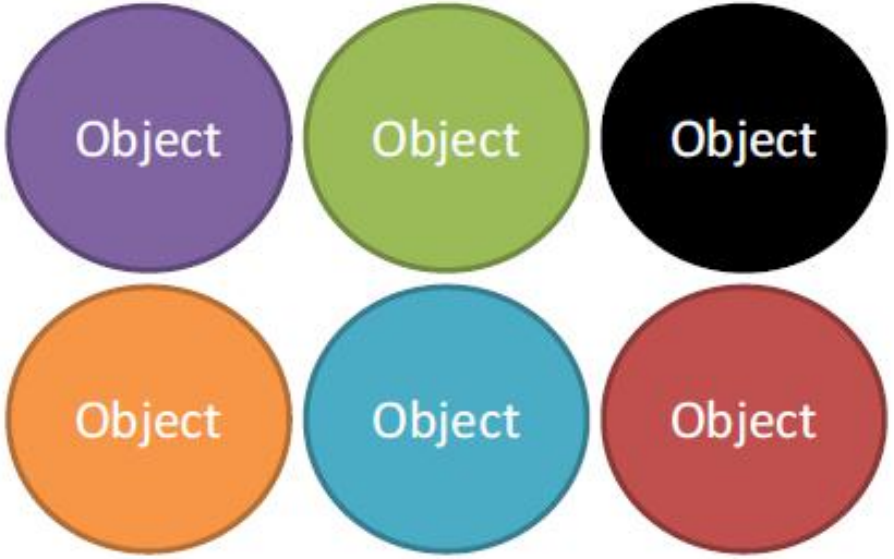
How java stores objects

- Objects are too big to fit in a variable
  - Stored somewhere else
  - Variable stores a number that locates the object

How java stores objects

- Objects are too big to fit in a variable
  - Stored somewhere else
  - Variable stores a number that locates the object





References

- The object's location is called a reference
- == operator compares the references

```
Point point1 = new Point(10,10);
Point point2 = new Point(10, 10);
```

Does point1 ==point2 ?

References

- The object's location is called a reference
- == operator compares the references

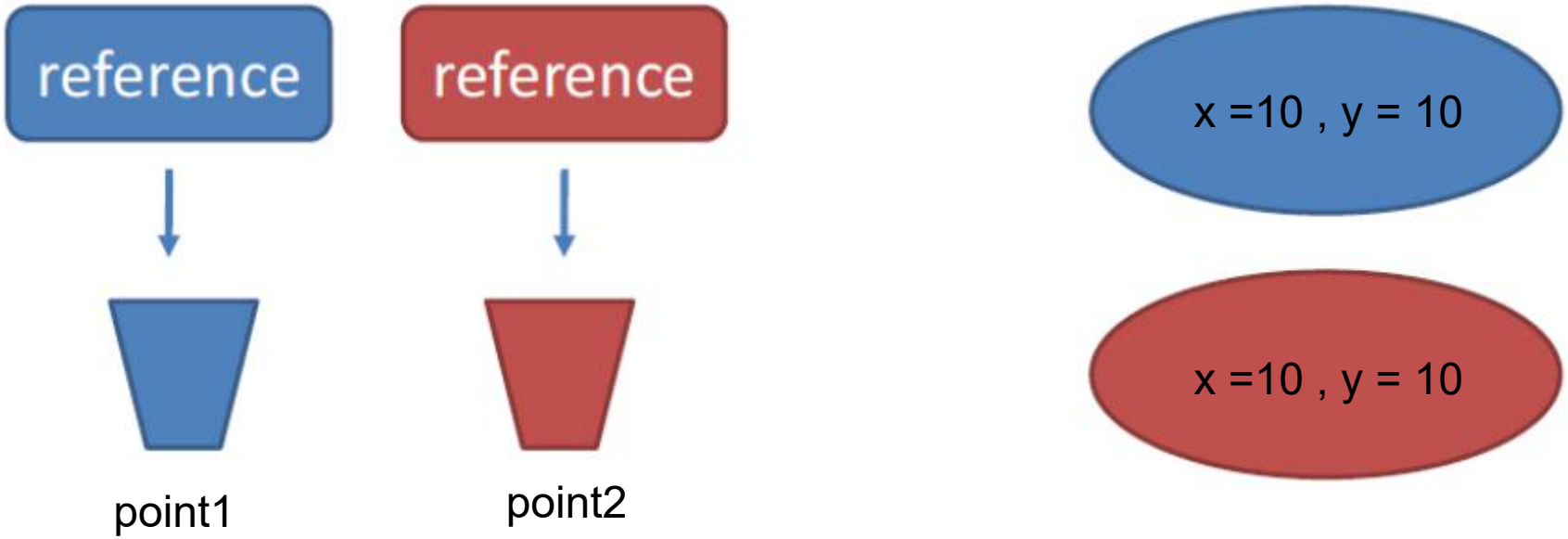
```
Point point1 = new Point(10,10);
Point point2 = new Point(10, 10);
```

Does point1 ==point2 ?

NO

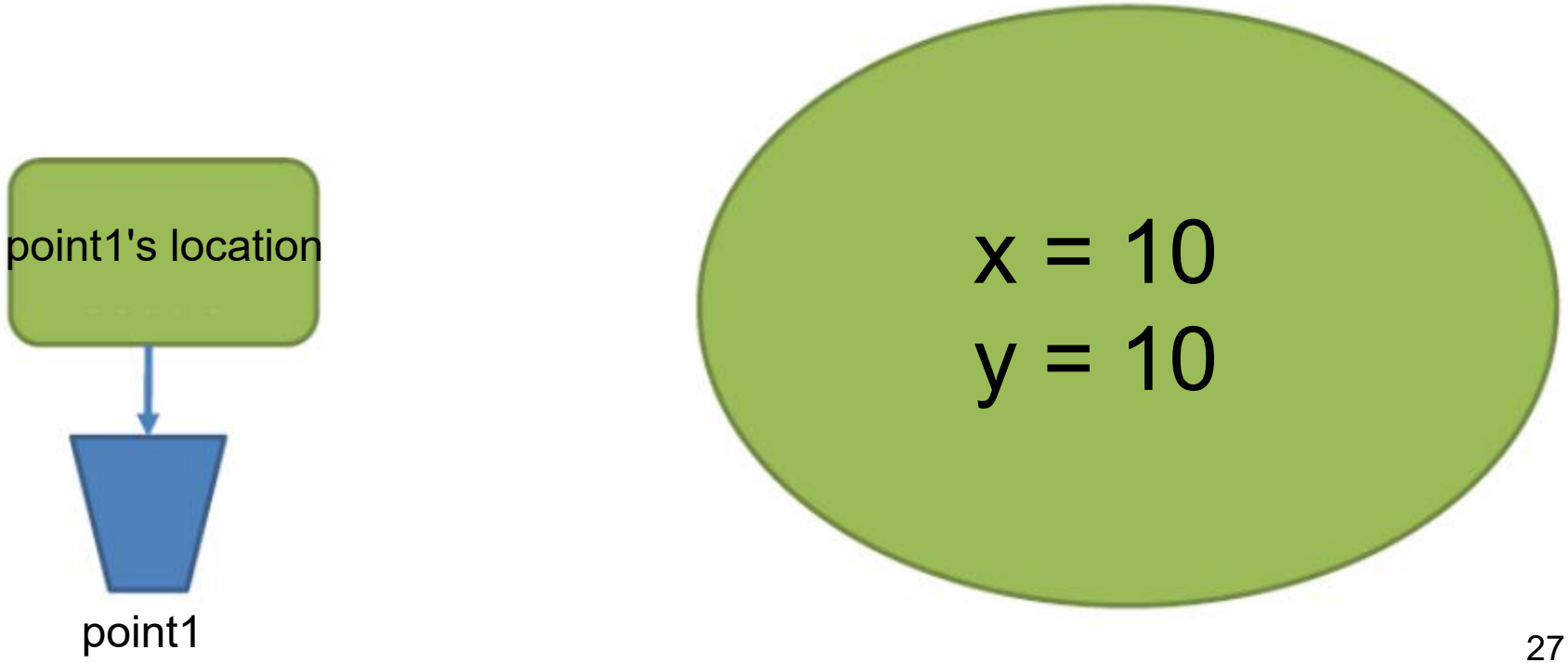
References

```
Point point1 = new Point(10,10);
Point point2 = new Point(10, 10);
```



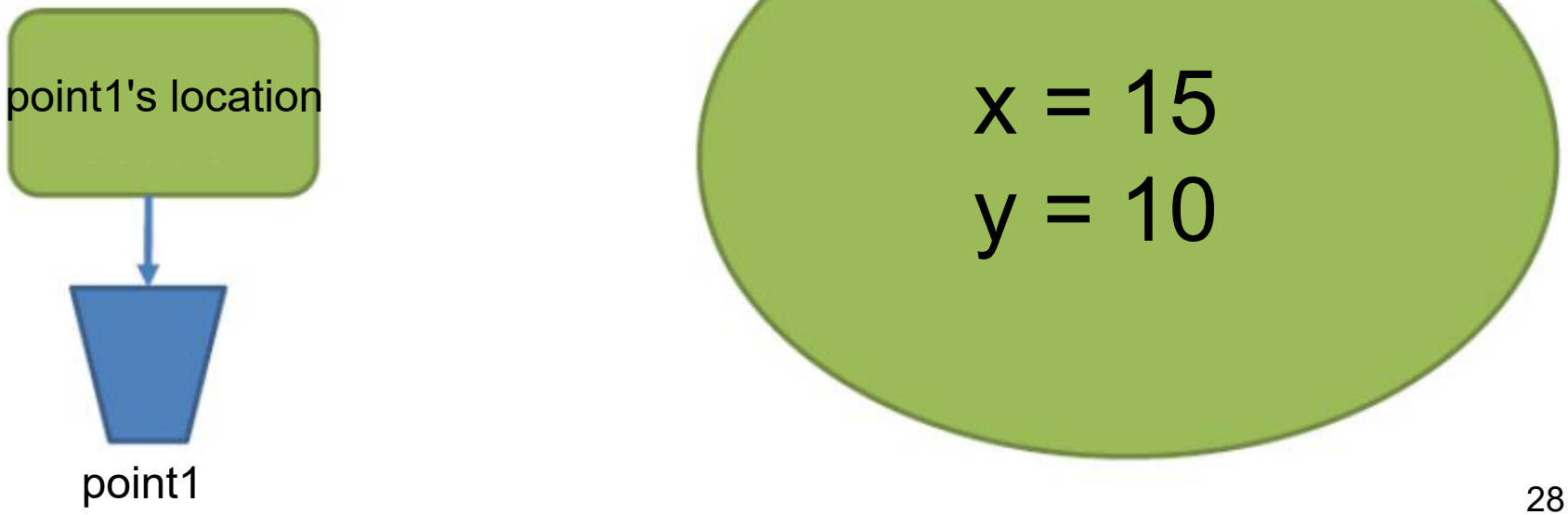
References

- Point point1 = new Point(10,10);



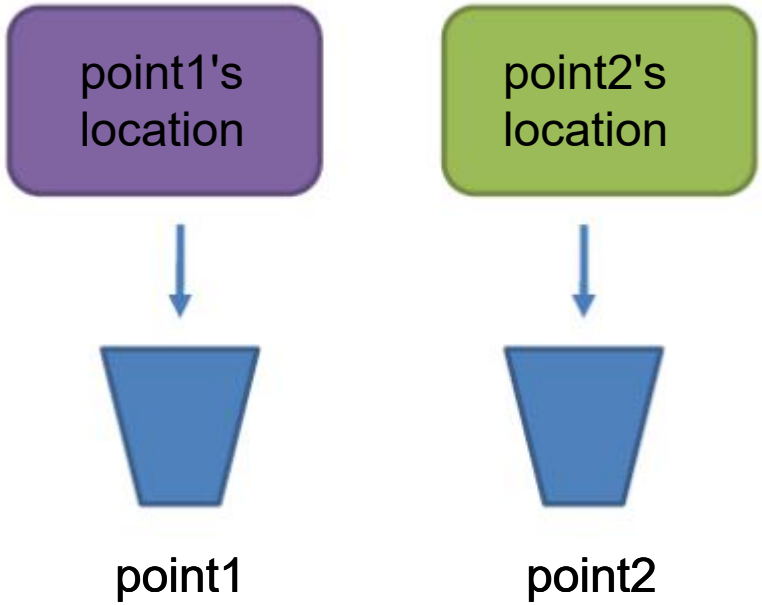
References

- Point point1 = new Point(10,10);
- point1.x = 15;



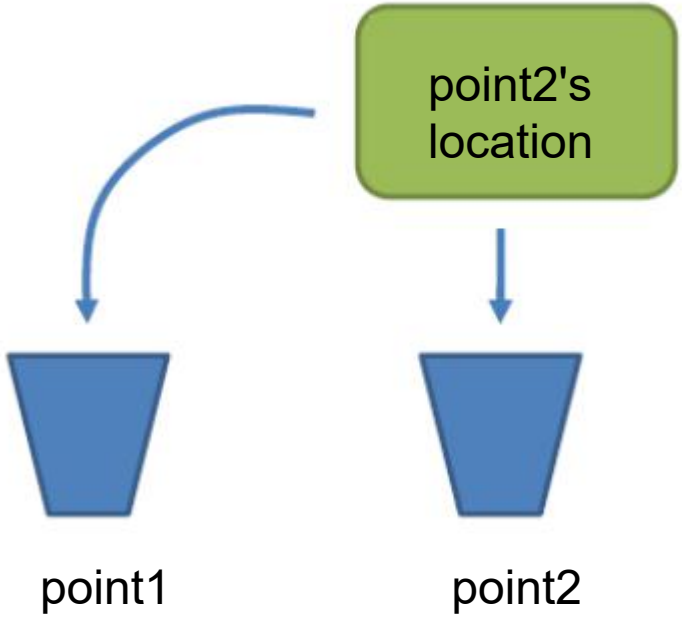
References

```
Point point1 = new Point(10,10);
Point point2 = new Point(10, 10);
```



References

point1 = point2;



static fields and methods

static field

- Applies to fields and methods
- Means the field/method
  - Is defined for the class declaration,
  - Is not unique for each instance

static field

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

        Point.count = 50;

        Point point1 = new Point(10,10);
        Point point2 = new Point(15, 22);

        Point.count = 5;

}
```

static field

- Keep track of the number of points

```
public class Point {

    int xCoordinate;
    int yCoordinate;

    static int count = 0;

    public Point(int x, int y){
        xCoordinate = x;
        yCoordinate = y;
    }


    public void move(int xDistance, int yDistance){
        xCoordinate += xDistance;
        yCoordinate += yDistance;
    }
}
```

static field

- Keep track of the number of points

```
public class Point {

    int xCoordinate;
    int yCoordinate;

    static int count;

    public Point(int x, int y){
        xCoordinate = x;
        yCoordinate = y;
        count ++;
    }


    public void move(int xDistance, int yDistance){
        xCoordinate += xDistance;
        yCoordinate += yDistance;
    }
}
```

References

- http://math.hws.edu/javanotes/
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/lecture-notes/
- https://docs.oracle.com/javase/tutorial/java
- https://courses.cs.washington.edu/courses/cse142/11au/lectures/11-23/23-ch08-3-encapsulation.ppt