SWINBURNE UNIVERSITY OF TECHNOLOGY

# COS70006 OOP

## Class & Object, and Application

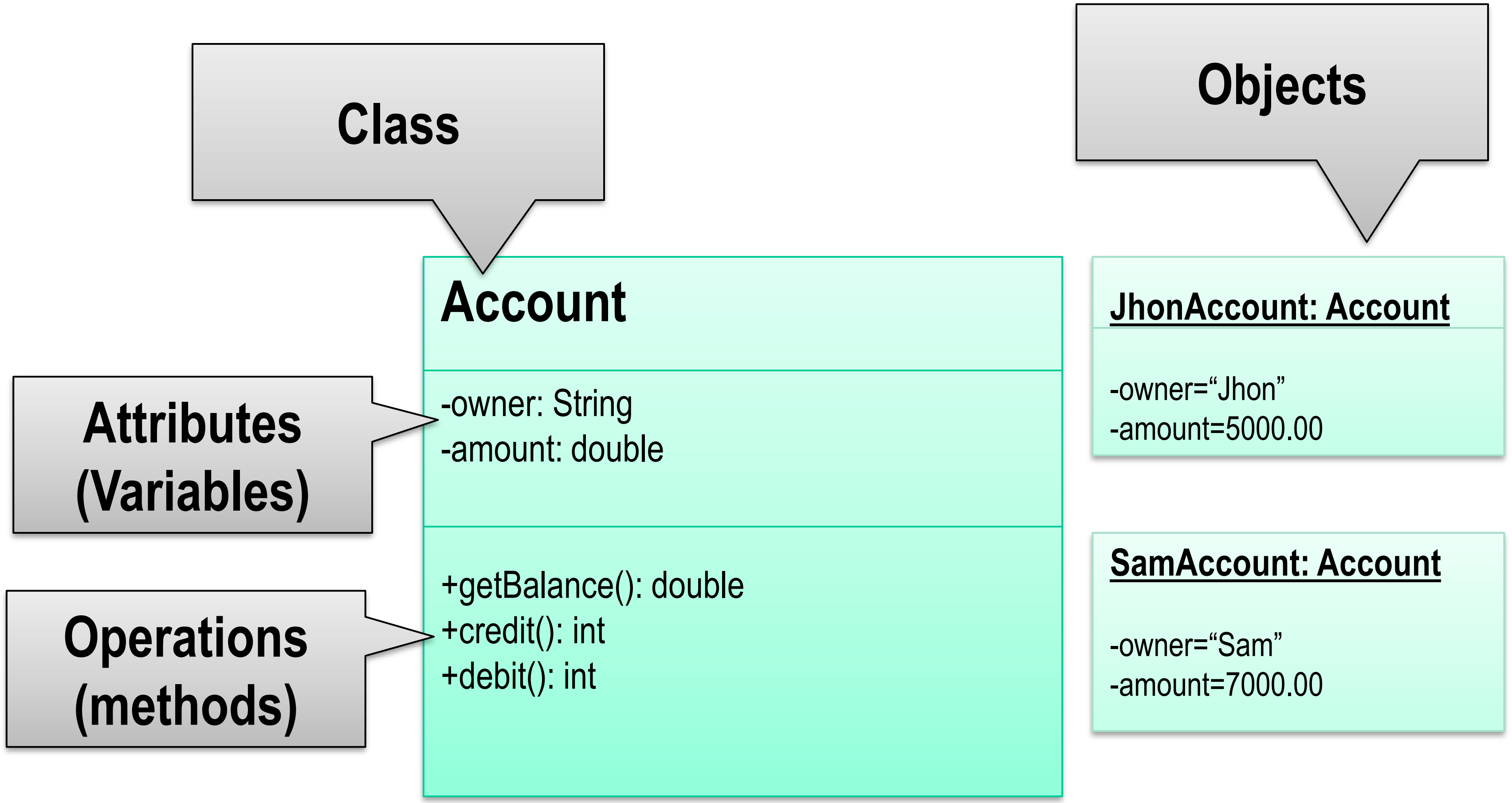
## Class & Object, and Application

- You should review lecture notes and examples from Week 1 to Week 6

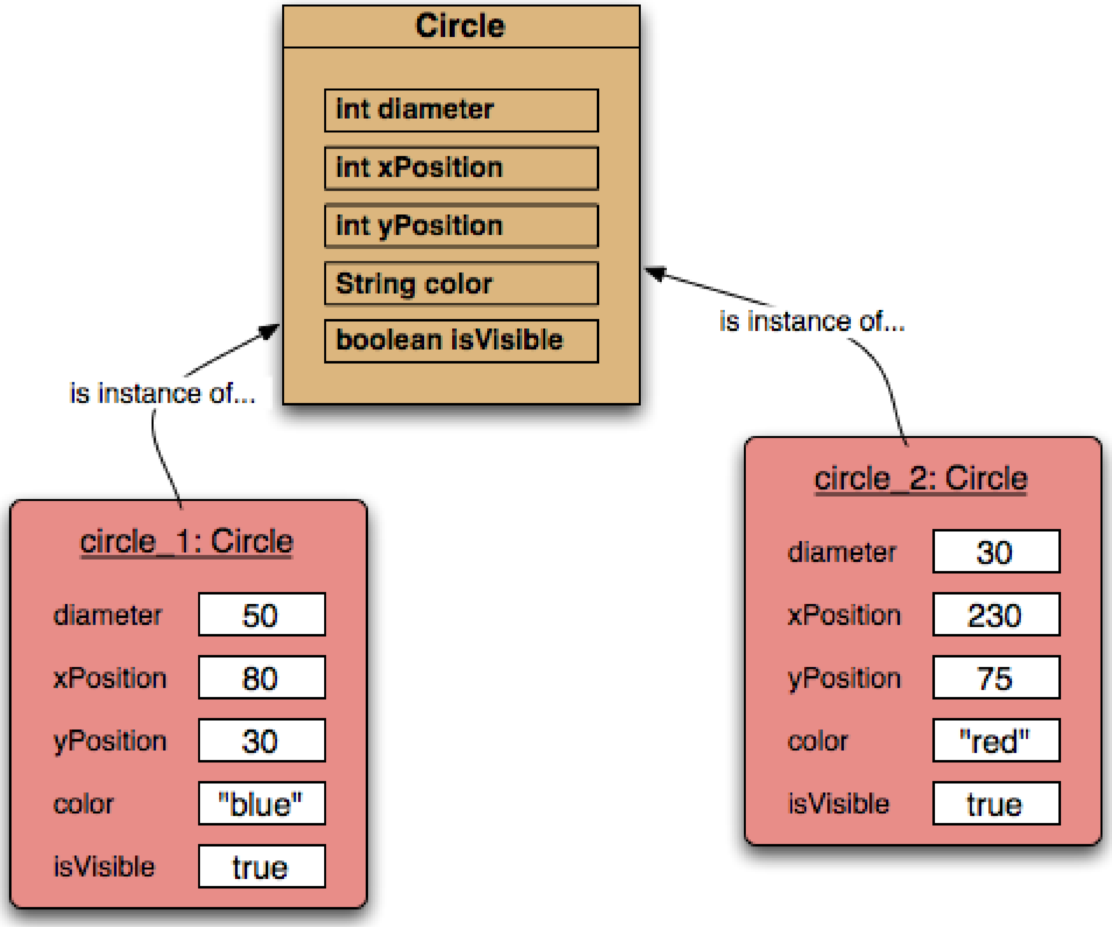
  e.g:

  - Java Basics.ppt (Week 1)
  - Object creation and collaboration.ppt (Week 2)
  - Principles of OOP.ppt (Week 4)

## UML notation ( a slide from Week 1)



## Two circle objects ( a slide from Week 1) (Instances of the Circle class)



Slide supplied by Barnes and Kolling

## Declaring a Java Class and using Objects

```
public class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```
import java.util.Math;

public class LineSeg {
    private Point begin;
    private Point end;

    public LineSeg(Point begin, Point end) {
        this.begin = begin;
        this.end = end;
    }

    public double getDistance() {
        int x1 = begin.getX();
        int y1 = begin.getY();
        int x2 = end.getX();
        int y2 = end.getY();
        double dist = Math.sqrt(
                (x2 – x1)^2 + (y2 – y1)^2 );
        return dist;
    }
}
```

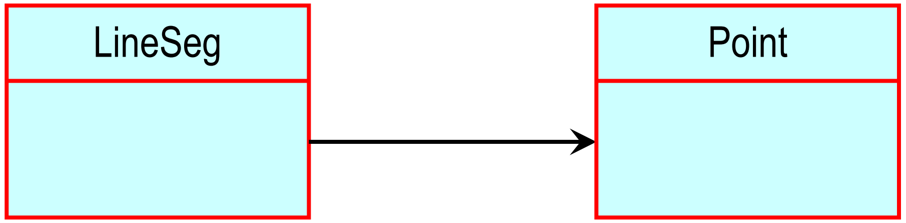
## Tying it altogether

```
// assuming Point.java, LineSeg.java and LineProgram.java are in the same folder

public class LineProgram {
    public static void main(String [] args) {
        Scanner in = new Scanner(System.in);
        int x = in.nextInt();
        int y = in.nextInt();
        Point pt1 = new Point(x, y);
        x = in.nextInt();
        y = in.nextInt();
        Point pt2 = new Point(x, y);
        LineSeg line = new LineSeg(pt1, pt2);
        System.out.println(“The distance is ” + line.getDistance());
    }
}
```

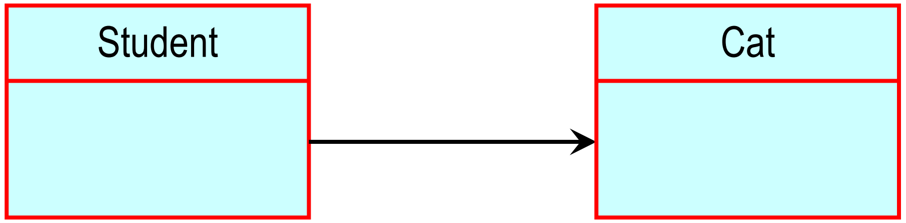
## Collaborating classes: Association

- Classes
  - LineSeg
  - Point
  - LineProgram – Appplication (with main)



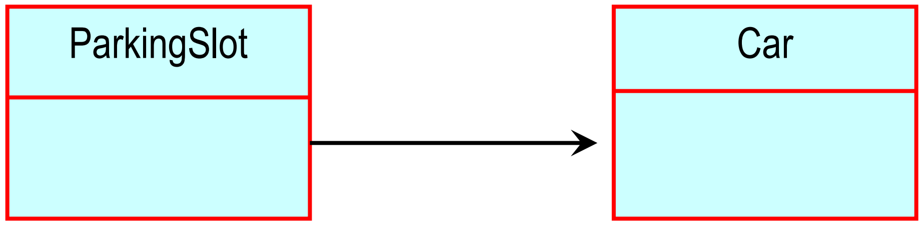
# Collaborating classes: Association

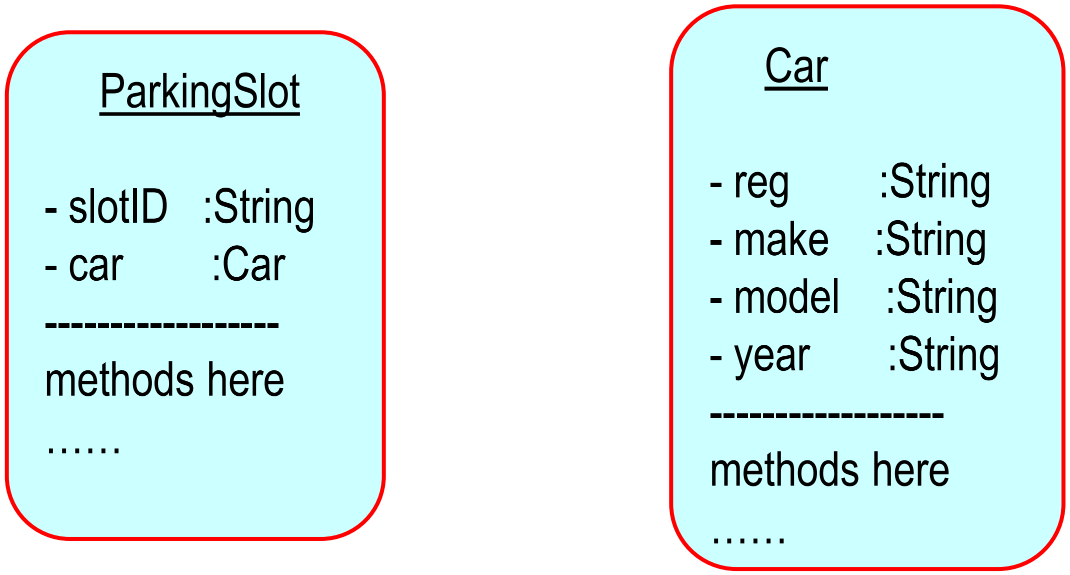
- We say there is an <u>association</u> between the two classes
- If Student calls methods in Cat then we say
  Cat is a <u>collaborator</u> of Student
  - "Cat helps Student"
  - or "Cat provides services for Student"
  - or "Cat is a <u>server</u> for Student"



# Collaborating classes: Association

- ParkingSlot & Car

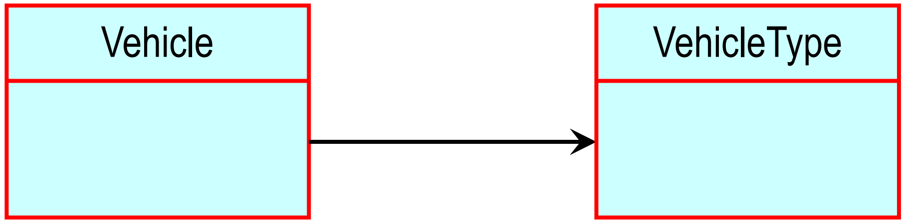




## An example: Vehicle & VehicleType

- Classes
  - Vehicle
  - VehicleType
  - VehicleHireApp – Appplication (with main)

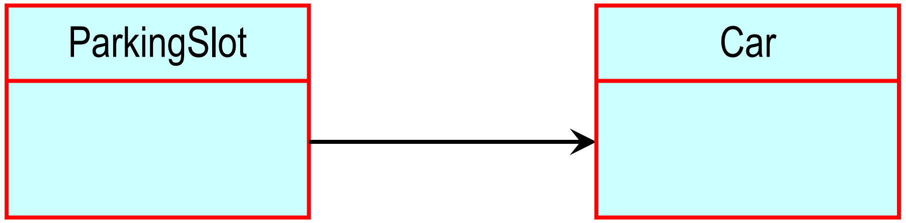
    use ArrayList<Vehicle> vehicles for a list of vechicles



## An approach: ParkingSlot & Car

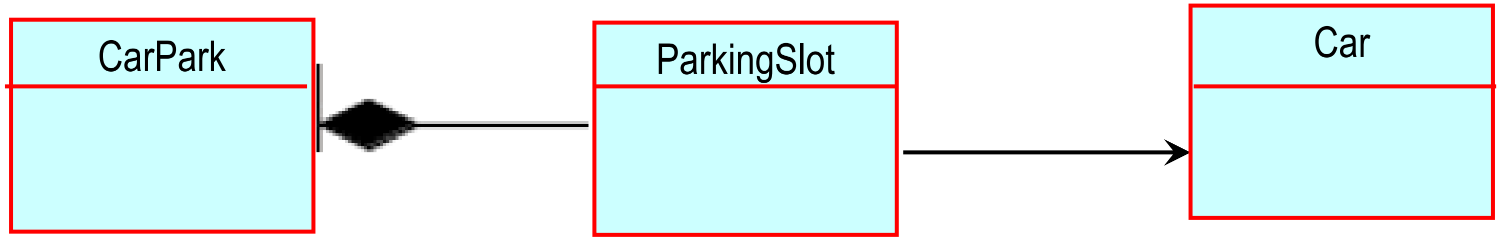
- Classes
  - ☐ ParkingSlot
  - ☐ Car
  - ☐ CarParkingApp – Appplication (with main)

    use ArrayList<ParkingSlot> for a list of ParkingSlots



## Another approach: ParkingSlot & Car

- Classes
  - ParkingSlot
  - Car
  - CarPark - use ArrayList<ParkingSlot> for a list of ParkingSlots with methods addSlot, removeSlot, …
  - CarParkingApp – Appplication (with main)



You might have a different approach …..