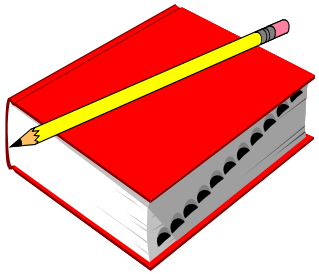
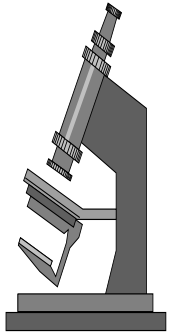
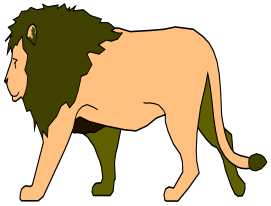
Object Oriented Programming

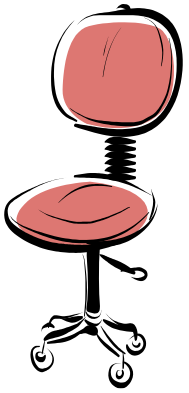
Real World Objects

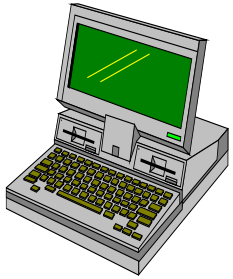






Real world entities





Real World Objects

Objects have attributes (state)...

ATTRIBUTES
Name : Pamuk
Color : White
Breed : White Terrier
Hungry: Yes

ATTRIBUTES
Current Gear: 4
Current Direction: West
Current Speed: 90 km/h
Color:White

Objects have behaviours

BEHAVIOUR
Barking
Fetching
Eating
Running

ATTRIBUTES
Name : Pamuk
Color : White
Breed : White Terrier
Hungry: Yes

BEHAVIOUR
Change Gear
Change Direction
Accelerate
Apply Brakes

ATTRIBUTES
Current Gear
Current Direction
Current Speed
Color

Objects have behaviours

ATTRIBUTES
On: Yes

BEHAVIOUR
Turn On
Turn Off

ATTRIBUTES
On: Yes
Current Volume: 5
Current Station: 103.1

BEHAVIOUR
Turn On
Turn Off
Increase Volume
Decrease Volume
Seek
Scan

Example: A “Rabbit” object

You could (in a game, for example) create an object representing a rabbit

It would have data:

How hungry it is

How frightened it is

Where it is

And methods:

eat, hide, run, jump

Classes (Categories)

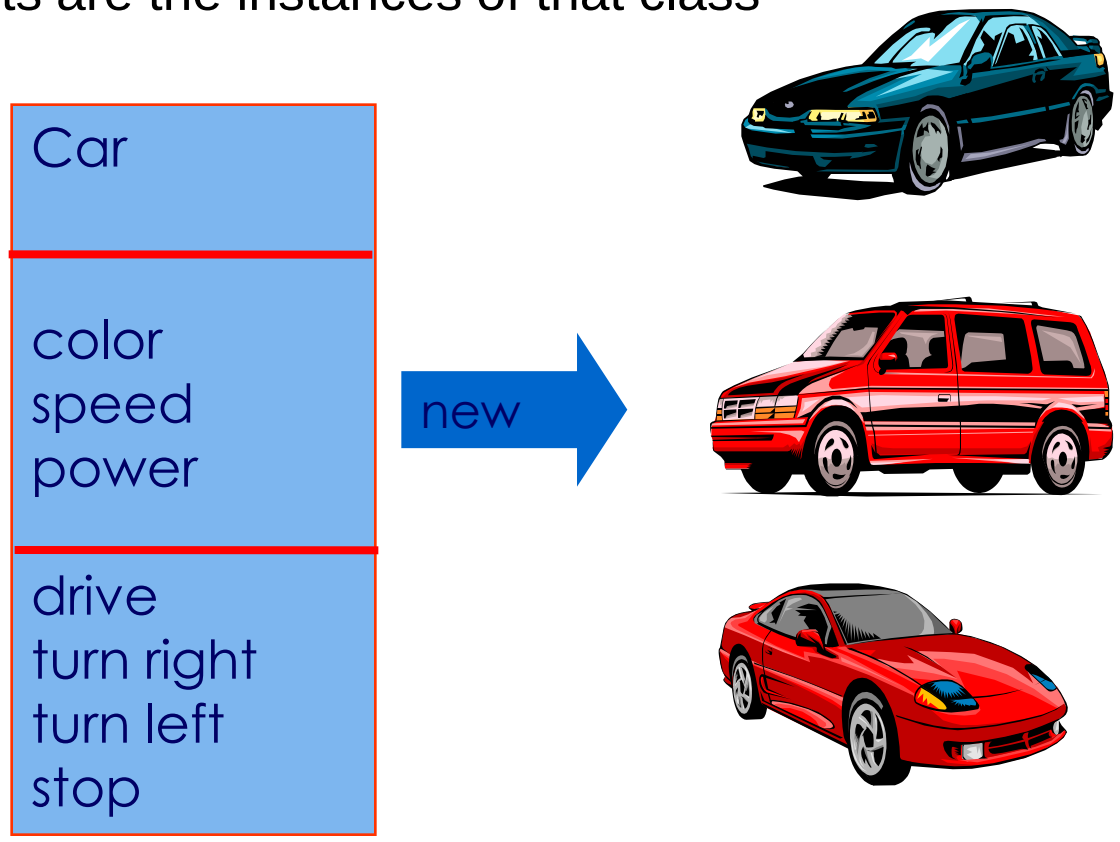
Classes

Serves as template/blueprint from which objects can be created

Defines **attributes** and **operations**

Can be used to *create* objects

Objects are the instances of that class



Example: Student

- Represent the real world

  Student

Example: Student

- Represent the real world

  Student

  name
  id
  year
  courses
  email

Example: Student

- Objects group together
  - Primitives (int, double, char, etc..)
  - Objects (String, etc…)

Student (class groups the following data)

String name
String id
int year
ArrayList courses
String email

Why use classes?

- Why not just primitives?

```
// student Ali
String nameAli;
int yearAli;

//student Mehmet
String nameMehmet
int yearMehmet;
```

Why use classes?

- Why not just primitives?

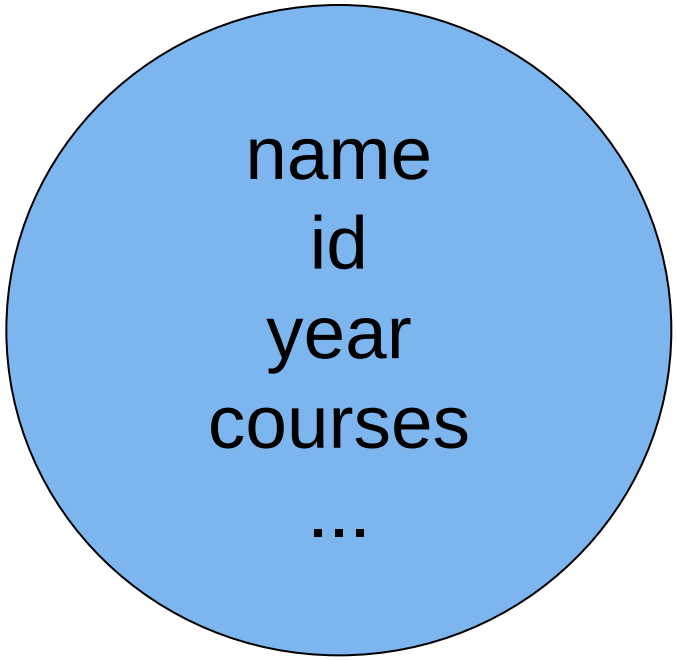
```
// student Ali
String nameAli;
int yearAli;

//student Mehmet
String nameMehmet
int yearMehmet;
```

When there are 200 Students ?

We have to repeat each property of Student 200 times!

Why use classes?

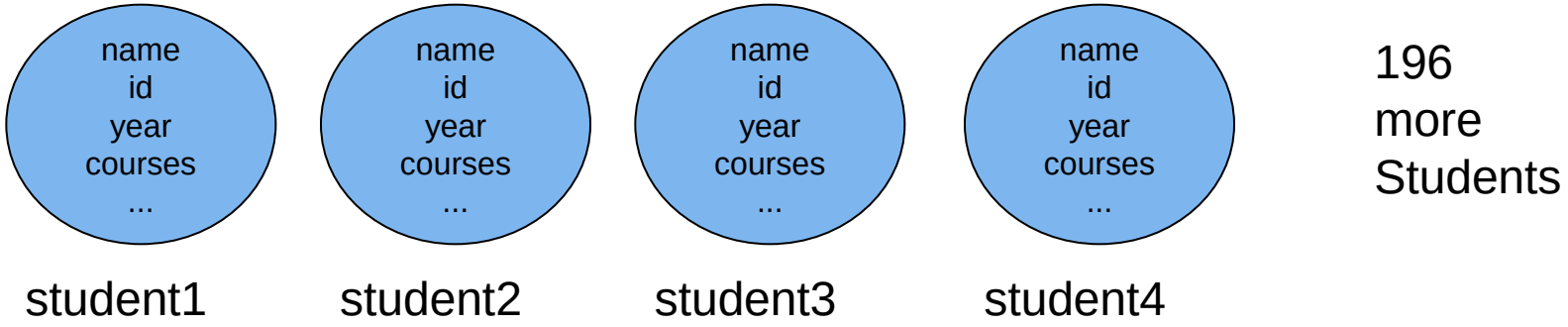
- With a class you can group Student's properties



student1

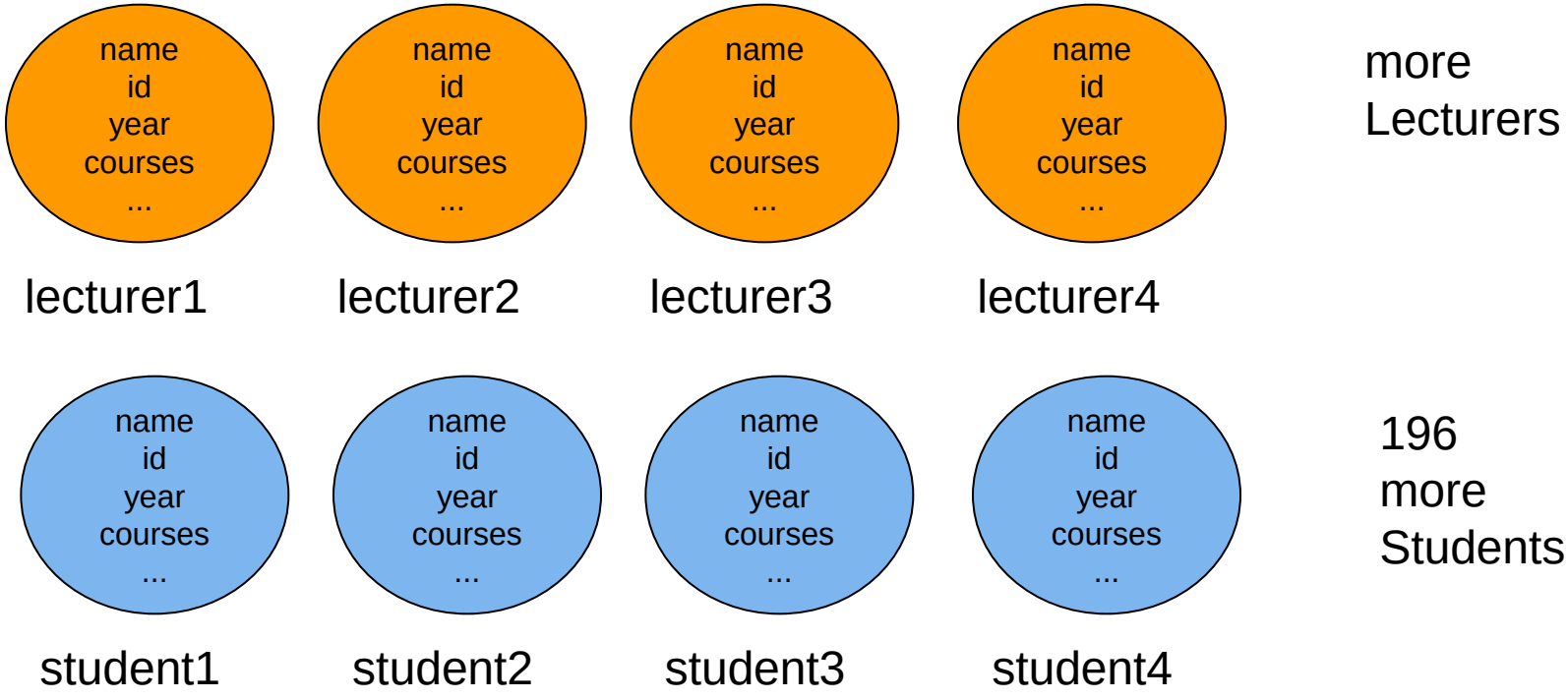
Why use classes?

- You can have multiple instances from Student class to represent each student



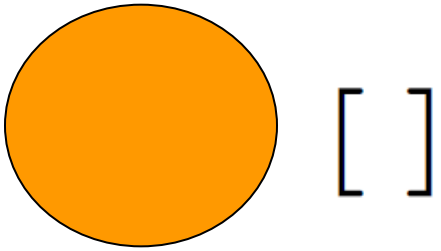
Why use classes?

- University (Student and Lecturer objects)

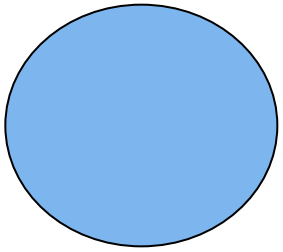


Why use classes?

University



Lecturer



Student

You can create arrays of Students and Lecturers in your program.

```
Lecturer[] lecturers = new Lecturer[10];
```

```
Student[] students = new Student[10];
```

```
Classes introduces new types to Java!
```

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!

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

    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);

}
```

Accessing Class Property

```
public class Point {

        private int xCoordinate;
        private int yCoordinate;

        public Point(int x, int y){
                setxCoordinate(x);
                setyCoordinate(y);
        }

        int getxCoordinate() {
                return xCoordinate;
        }
```

Continue: Accessing Class Property

```
    void setxCoordinate(int xCoordinate) {
            this.xCoordinate = xCoordinate;
    }

    int getyCoordinate() {
            return yCoordinate;
    }

    void setyCoordinate(int yCoordinate) {
            this.yCoordinate = yCoordinate;
    }

}
```

Accessing fields by Get Methods

- 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. getxCoordinate()
                + ", y: " + point1.getyCoordinate());

    }

}
```

Accessing fields by Set Methods

- Object.FIELDNAME

```
public class Test {

    public static void main(String[] args) {
            Point point1 = new Point(10,10);
            Point point2 = new Point(15, 22);
             point1. setxCoordinate(4);
            point1. setyCoordinate(3);

            System.out.println("x: " + point1. getxCoordinate()
                        + ", y: " + point1.getyCoordinate());
    }

}
```

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://www.tiobe.com/tiobe-index/
- Head First Java 2nd Edition
- Java a Beginner's guide 7th Edition