Advanced Programming Techniques in Java

Review: 2D Arrays

- **Declaring** and **creating** a 2D array:

```
<type> [][]arrayName = new <type> [<r
```

```
int[][] score = new int[5][8];
```

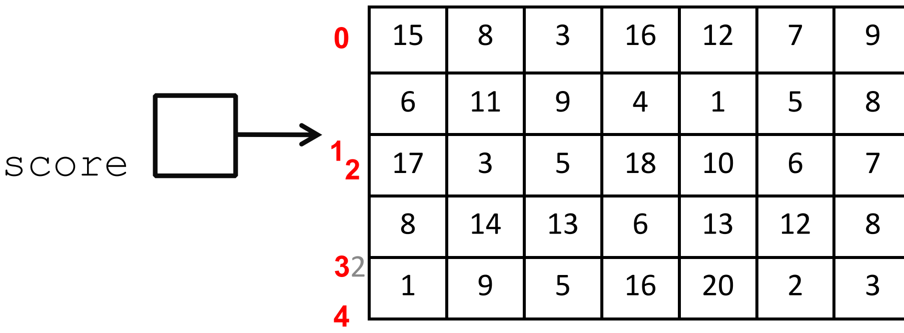
Number of rows

Nu

- To **access** an element: `arrayName[<row>][<c`

`score[3][4]` will give you the value at row 3, colun

0 1 2 3 4 5 6



15	8	3	16	12	7	9
6	11	9	4	1	5	8
17	3	5	18	10	6	7
8	14	13	6	13	12	8
1	9	5	16	20	2	3

Review: Wrapper classes

Primitive Type	**Wrapp**
`int`	`Integ`
`double`	`Doubl`
`char`	`Chara`
`float`	`Float`

boolean	Boole

- A wrapper is an object whose sole purpose is to hold
- Once you construct the list, use it with primitives as n

Review: Bubble So

```
public static void bubbleSort(int[]
   arr){ int didswap = 1, tmp = 0; w
   (didswap == 1) { didswap = 0;
       for (int i = 1; i < arr.lengt
             { if (arr[i - 1] > arr
             tmp = arr[i - 1]; arr[i
             arr[i]; arr[i] = tmp; d
             1;
             }
       }
   }
}
```

Class objectives

- Intro to Object Oriented Design (Section 8.1?

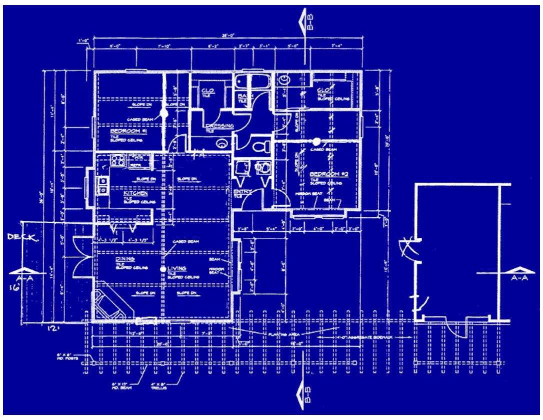
Classes and Ob

Review: State and Behavior

- **Definition**
- A **state** is a set of values (internal data) stored in ar
- **Definition**
- A **behavior** is a set of actions an object can perforn internal state

Review: What is a Class?

- It is a definition of a new type of objects



- The objects of a given class are built according to it
- Objects of a class are referred to as instance of the

- **Definition**
- A **class** is like a blueprint (defined by the user) for w

Review: Object State: Fields

- **Definition**
- A field is a variable inside an object that makes up part o

- **Syntax**

```
<type> <name>;
```

- **Example** `public class Student{` Each `Student`

```
public class Student{
     double gpa;
}
```

`gpa` field

`String name;`

Review: Constructing objects

- **Construct**: To create a new object
- Objects are constructed with the **`new`** keyword
- Most objects must be constructed before they can be us
- **Syntax**
- Strings are also objects, but can be constructed wi

```
String name = "Amanda Ann Camp";
```

```
<type> <name> = new <type> ( <parame
```

- **Example**:

```
Point p = new Point();
```

Review: `Point` Class (ver. 1)

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

- The `Point` class isn't itself an executable program

  themselves are not complete programs
- They can only be used as part of larger programs to solve
- The program that creates and uses objects is known
  code

Client Code

- **Definition**
- A **client code** is the code that interacts with a class or obj

- The way a client code interacts with the objects is by asking them to perform behavior

- Remember the objects from the built-in classes `Str`
- You and your programs have been clients of these objects

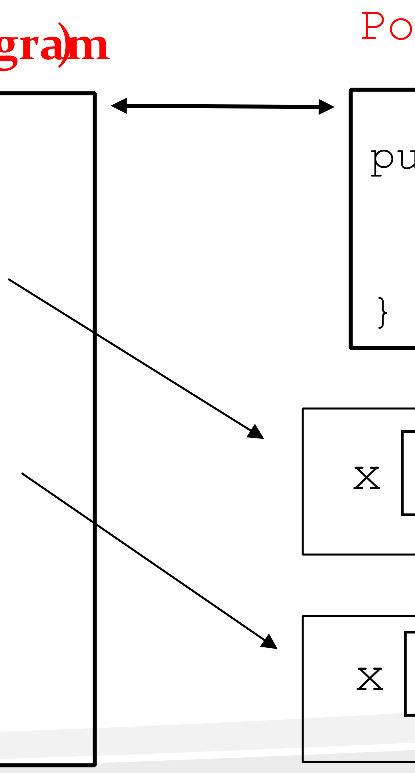
A Class and its Client

- A class can be used by client programs

PointMain.java **(client program)**

```
public class PointMain {
   ... main(String[] args) {
      Point p1 = new Point();
      p1.x = 7;
      p1.y = 2;

      Point p2 = new Point();
      p2.x = 4;
      p2.y = 3;
      ...
   }
}
```



- `Point.java` is not, by itself, a runnable program

```
public class PointMain {
    public static void main(String[] args)
        // create two Point objects
        Point p1 = new Point();
        p1.y = 2;
        Point p2 = new Point();
        p2.x = 4;

        // print p1
        System.out.println(p1.x + "," + p1.

        // move p2 and then print it
        p2.x += 2;
        p2.y++;
        System.out.println(p2.x + "," + p2.
    }
}
```

```
public class PointMain {
    public static void main(String[] args) {
        // create two Point objects
        Point p1 = new Point();
        p1.x = 7;
        p1.y = 2;

        Point p2 = new Point();
        p2.x = 4;
        p2.y = 3;

         // print the points
        System.out.println("p1 is (" + p1.x + "," +
        System.out.println("p2 is (" + p2.x + "," +

        // translate each point to a new location
        p1.x += 11;
        p1.y += 6;
        p2.x += 1;
        p2.y += 7;

        // print the points
        System.out.println("p1 is (" + p1.x + "," +
        System.out.println("p2 is (" + p2.x + "," +
    }
}
```

Client Program for the `Point`

```
public class PointMain {
    public static void main(String[] args) {
        // create two Point objects
        Point p1 = new Point();
        p1.x = 7;
        p1.y = 2;

        Point p2 = new Point();
        p2.x = 4;
        p2.y = 3;

           // print the points
        System.out.println("p1 is (" + p1.x + "," + p1.y + ")");
        System.out.println("p2 is (" + p2.x + "," + p2.y + ")");

        // translate each point to a new location
        p1.x += 11;
        p1.y += 6;
        p2.x += 1;
        p2.y += 7;

        // print the points
        System.out.println("p1 is (" + p1.x + "," + p1.y + ")");
        System.out.println("p2 is (" + p2.x + "," + p2.y + ")");
    }
}
```

Not the best way

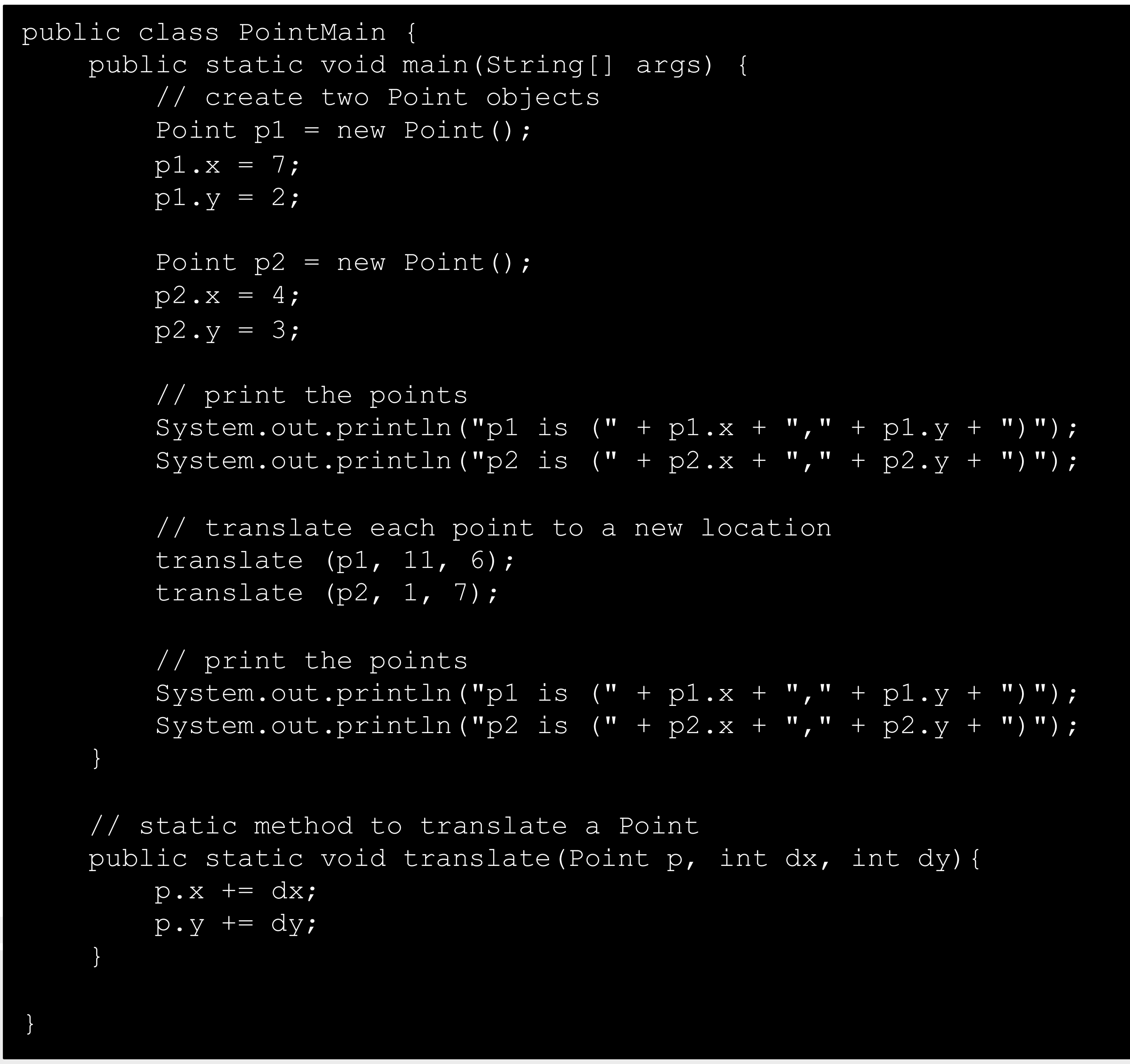
Client Program for the `Point` Cl

- The client program has some redundancy
- Translating a point is a common operation, we shou

```
//A static method to translate a Point
public static void translate(Point p, int
       p.x += dx;
       p.y += dy;
}
```

```
//Method call
translate(p1, 11, 6)
```

Client Program for the Point C



```
public class PointMain {
    public static void main(String[] args) {
        // create two Point objects
        Point p1 = new Point();
        p1.x = 7;
        p1.y = 2;

        Point p2 = new Point();
        p2.x = 4;
        p2.y = 3;

        // print the points
        System.out.println("p1 is (" + p1.x + "," + p1.y + ")");
        System.out.println("p2 is (" + p2.x + "," + p2.y + ")");

        // translate each point to a new location
        translate (p1, 11, 6);
        translate (p2, 1, 7);

        // print the points
        System.out.println("p1 is (" + p1.x + "," + p1.y + ")");
        System.out.println("p2 is (" + p2.x + "," + p2.y + ")");
    }

    // static method to translate a Point
    public static void translate(Point p, int dx, int dy){
        p.x += dx;
        p.y += dy;
    }

}
```

Problem with Static Methods

- We are missing a major benefit of objects: code reus
- Every program that uses `Point` objects would need a tra
- So far, our `Point` class contains state, but no beha
- The reason of classes is to combine state and behav
- The `translate` method belong inside each `Point` obje

Instance Met

Object Behavior: Methods

- **Definition**
- An **instance method** (or **object method**) is a meth
  class and gives behavior to each object
- **Syntax** `public <type> <name>(<type> <name>,`

```
 }
```

- **Example**

```
public void shout() {
      System.out.println("HELLO THERE!");
}
```

- Same syntax as static methods, but without `statio`

Point Class (ver. 2)

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

   // shifts this point;s location by the
   amount public void translate(int dx, i
   dx; y += dy;
  }
}
```

- The translate method no longer has a `Point p` para
  - How does the method know which point to translate?
  - How does the method access the point’s `x` and `y` data

Calling Instance Methods

- Objects contain methods that can be called by your
- When we call an object's method, we are sending a mess
- We must specify which object we are talking to, and then
- **Syntax**

  <object name>.<method name>(<parameters)>

  The result will be different from one object to another
- **Example**

```
String s1 = ”Iraklis";
String s2 = "Antonella";

System.out.println(s1.length());     // 7
System.out.println(s2.length());     // 9
```

toString()

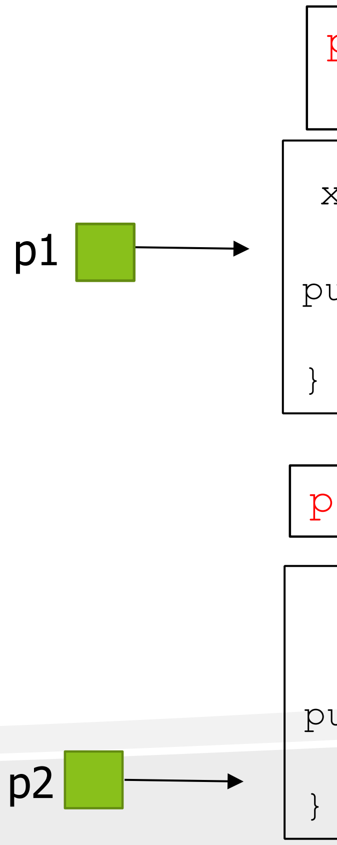
Point Objects with Method

- Each Point object has its own copy of the transla
object’s state

```
Point p1 = new Point();
p1.x = 7;
p1.y = 2

Point p2 = new Point();
p2.x = 4;
p2.y = 3;

p1.translate(11, 6);
p2.translate(1, 7);
```



Implicit Parameter

- **Definition**
- An **implicit parameter** is the object on which an ins

```
p1.translate(11, 6)
```

- The object referred to by `p1` is the implicit paramete
- During the call `p2.translate(1, 7)`
- The object refereed to by `p2` is the implicit paramet

Printing Objects

- By default, Java doesn't know how to print objects

```
Point p = new Point();
p.x = 10;
p.y = 7;
System.out.println("p is " + p);  //p is P

//better, but cumbersome     p is (10, 7)
System.out.println("p is (" + p.x + ", " +

//desired behavior
System.out.println("p is " + p);  //p is (
```

The toString Method

- When a Java program is printing an object or conca special method called **toString()**
- The toString() method tells Java how to conve

```
Point p1 = new Point(7, 2);
System.out.println("p1: " + p1);

//the above code is really calling the foll
+ p1.toString());
```

- Every class has a toString method, even if it’s n

- Default: class's name @ object's memory addres

The `toString` Method (cont.

- **Syntax** `public String toString() {`

        code that returns a String representing this object;

  ```
  }
  ```

- **Example**

  ```
  //Returns a String representing this
  Point public String toString() { return
  "(" + x + ", " + y + ")";
  }
  ```

- Method name, return, and parameters must match

The `toString` Method (cont.)

```
public class
     Point{ int x;
     int y;
     …
     …
     public String toString(){ return
     "(" + x + ", " + y + ")";     }
```

```
…
int i = 42;
String s = "hello";
Point p = new Point();

System.out.println("i is " + i);
SysSystem.out.println("s is " + s);
System.out.println("p is " + p);
…
```

Client

The `toString` Method Facts

- It is recommended to write a `toString()` method
- Do not place `println` statements in the `toStrin`
- `toString()` simply return a String that the client can us
- Keep in mind that well formed classes of objects do all

Point Class (ver. 3)

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

    // shifts points location by the given amou
    public void translate (int dx, int dy){
        x += dx;
        y += dy;
    }

    // toString method
    public String toString(){
        return "(" + x + " , " + y + ")";
   }
}
```

PointMain.java(ver. 3)

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

        // Create two Point objects
        Point p1 = new Point();
        p1.x = 7;
        p1.y = 2;

        Point p2 = new Point();
        p2.x = 4;
        p2.y = 3;


        // Translate p1
        p1.translate(11, 6);
        System.out.println("p1 is " + p1);
    }
}
```

Construct

Object Initialization

- To use a variable (of either primitive or reference typ
  name

- **Example** `int x;`

  ```
  Point p;
  ```

- Before you use a variable (of either primitive or refe

- Currently it takes 3 lines to create a Point and initial

```
Point p = new Point();
p.x = 3;
p.y = 8;        //tedious
```

Object Initialization (cont.)

- We'd rather specify the fields' initial values at the st

```
Point p = new Point(3, 8);    //b
```

- Such statement is not legal for our `Point` class, b specifies how to create a point with initial (x, y) loca

Constructor

- **Definition**

- A **constructor** initialize the state of a new obje

- **Syntax**

```
public <class name>(<type> <name>, …, <typ
             statement(s);
}
```

- **Example**

Constructor

- The constructor run when the client uses the `new`
- No return type is specified, it implicitly "returns" the

```
//Constructs a new point with given loca
public Point(int initialX, int initialY)
          x = initialX;
          y = initialY;
}
```

- If a class has no constructor, Java supplies a c
- The default constructor initialize all fields to zero-eq

```
public <class name>(<type> <name>, …, <typ
           statement(s);
}
```

Point Class (ver. 4) with Const

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

    // constructs a new point with the
    public Point(int initialX, int init
          x = initialX;
          y = initialY;
    }


    // shifts points location by the gi
     public void translate (int dx, int
          x += dx;
          y += dy;
     }

     // toString method
     public String toString(){
          return "(" + x + " , " + y + "
    }
}
```

same as the class's

Constr

Once
Java v

PointMain.java (ver. 4)

```
public class PointMain {
   public static void main(String[] args){
        //Create two Point objects
        Point p1 = new Point(5, 2);
        Point p2 = new Point(4, 3);
        //Print each point
        System.out.println("p1 is "+ p1);
        System.out.println("p2 is "+ p2);

        //Translate each point to a new location

        p1.translate(11, 6);
        p2.translate(1, 7);
        //Print the points again
        System.out.println("p1 is "+ p1);
        System.out.println("p2 is "+ p2);
   }
}
```