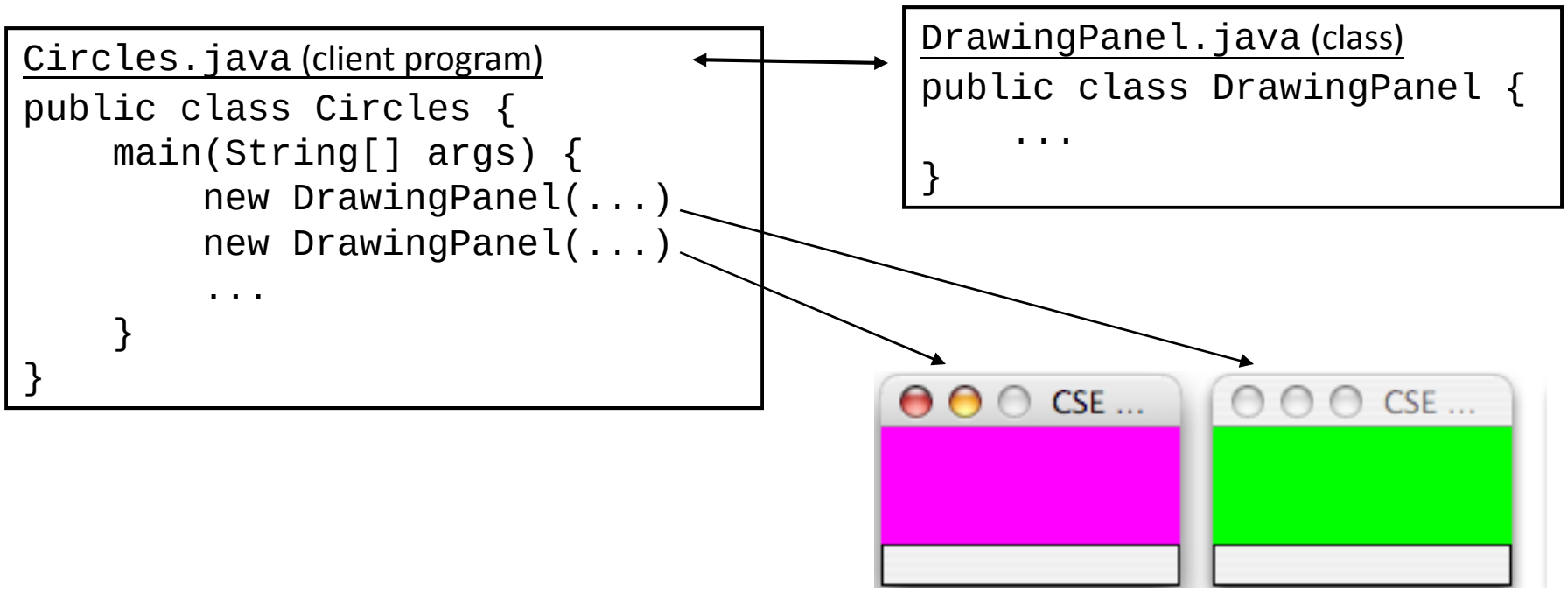
Garfield AP Computer Science

Classes

Thanks, Marty Stepp and Stuart Reges!! Most materials adapted from theirs.

Clients of objects

- **client program**: A program that uses objects.
  - Example: `Circles` is a client of `DrawingPanel` and `Graphics`.



Where do objects come from?

- **class**: A program entity that represents either:
  1. A program / module, or
  2. **A template for a new type of objects.**

  – The `DrawingPanel` class is a template for creating `DrawingPanel` objects.

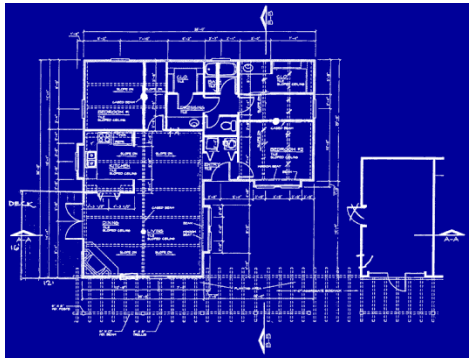
- **object**: An entity that combines state and behavior.
  – **object-oriented programming (OOP)**: Programs that perform their behavior as interactions between objects.

Blueprint analogy

iPod blueprint

state:
- current song
- volume
- battery life

behavior:
- power on/off
- change station/song
- change volume
- choose random song



creates

iPod #1

state:
- song = "1,000,000 Miles"
- volume = 17
- battery life = 2.5 hrs

behavior:
- power on/off
- change station/song
- change volume
- choose random song

iPod #2

state:
- song = "Letting You"
- volume = 9
- battery life = 3.41 hrs

behavior:
- power on/off
- change station/song
- change volume
- choose random song

iPod #3

state:
- song = "Discipline"
- volume = 24
- battery life = 1.8 hrs

behavior:
- power on/off
- change station/song
- change volume
- choose random song

`Point` objects (desired)

```
Point p1 = new Point(5, -2);
Point p2 = new Point();          // origin, (0, 0)
```

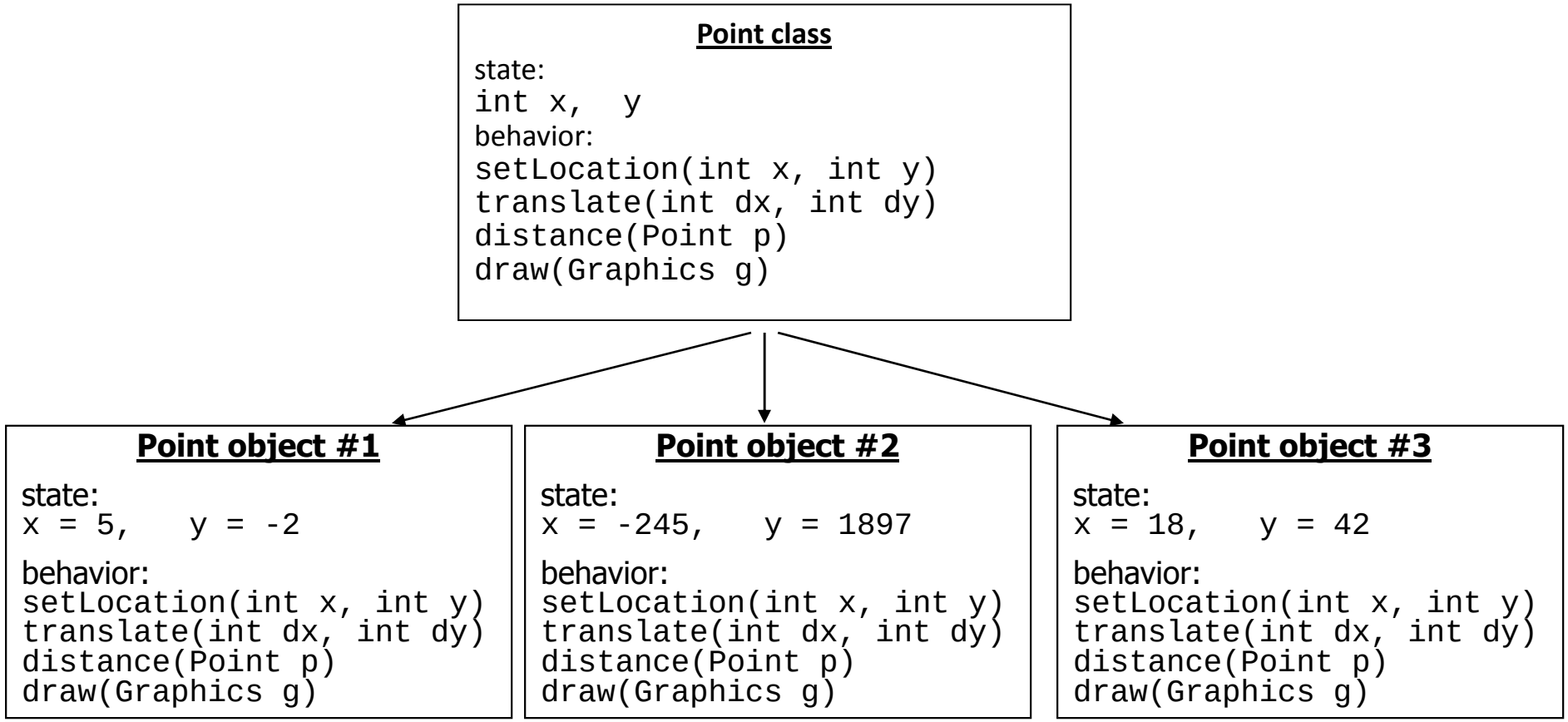
- Data in each `Point` object:

Field name	Description
x	the point's x-coordinate
y	the point's y-coordinate

- Methods in each `Point` object:

Method name	Description
`setLocation(`**`x`**`, `**`y`**`)`	sets the point's x and y to the given values
`translate(`**`dx`**`, `**`dy`**`)`	adjusts the point's x and y by the given amounts
`distance(`**`p`**`)`	how far away the point is from point *p*

`Point` class as blueprint



- The class (blueprint) describes how to create objects.
- Each object contains its own data and methods.

Fields

- **field**: A variable inside an object that is part of its state.
  - Each object has *its own copy* of each field.

- Declaration syntax:

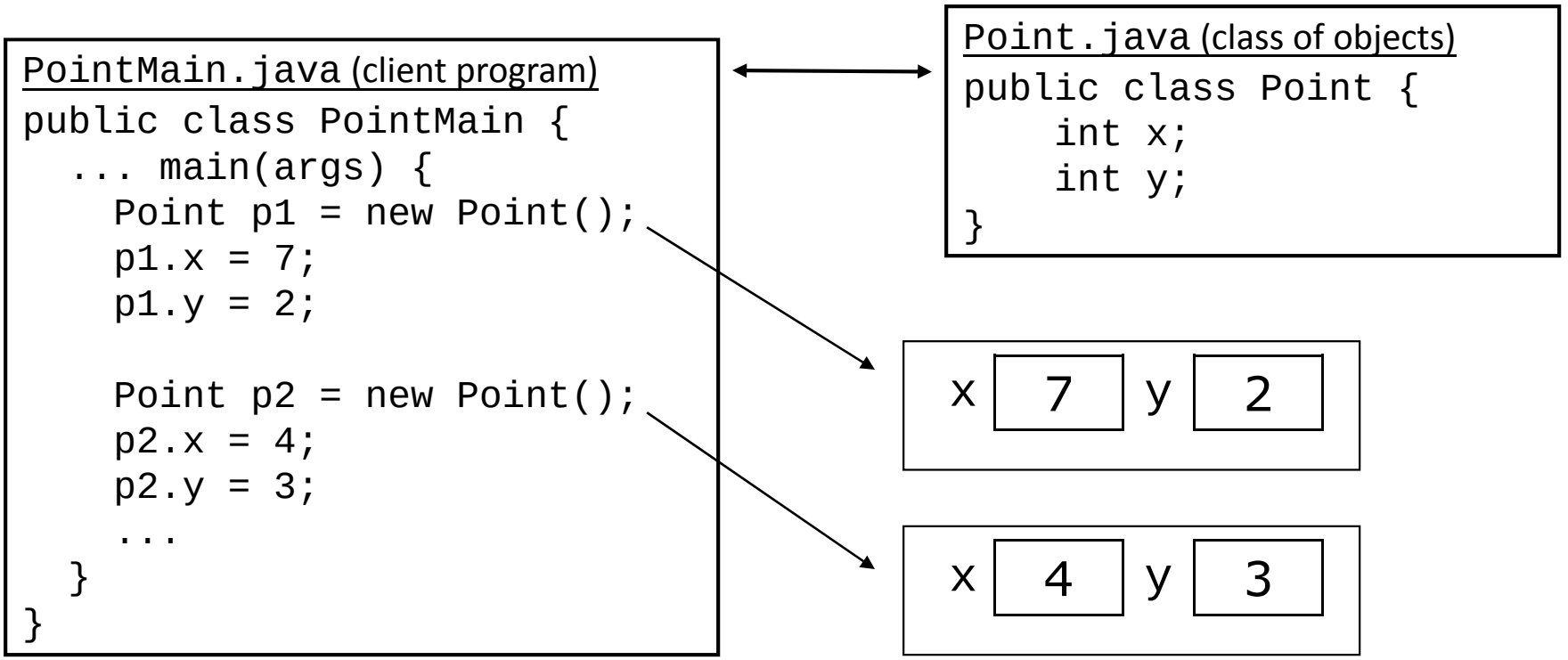
```
type name;
```

  - Example:

```
public class Student {
    String name;    // each Student object has a
    double gpa;     // name and gpa field
}
```

A class and its client

- `Point.java` is not, by itself, a runnable program.
  - A class can be used by client programs.



Instance methods

- **instance method**: One that exists inside each object of a class and defines behavior of that object.

```
public type name(parameters) {
    statements;
}
```

  - same syntax as static methods, but without `static` keyword

Example:

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

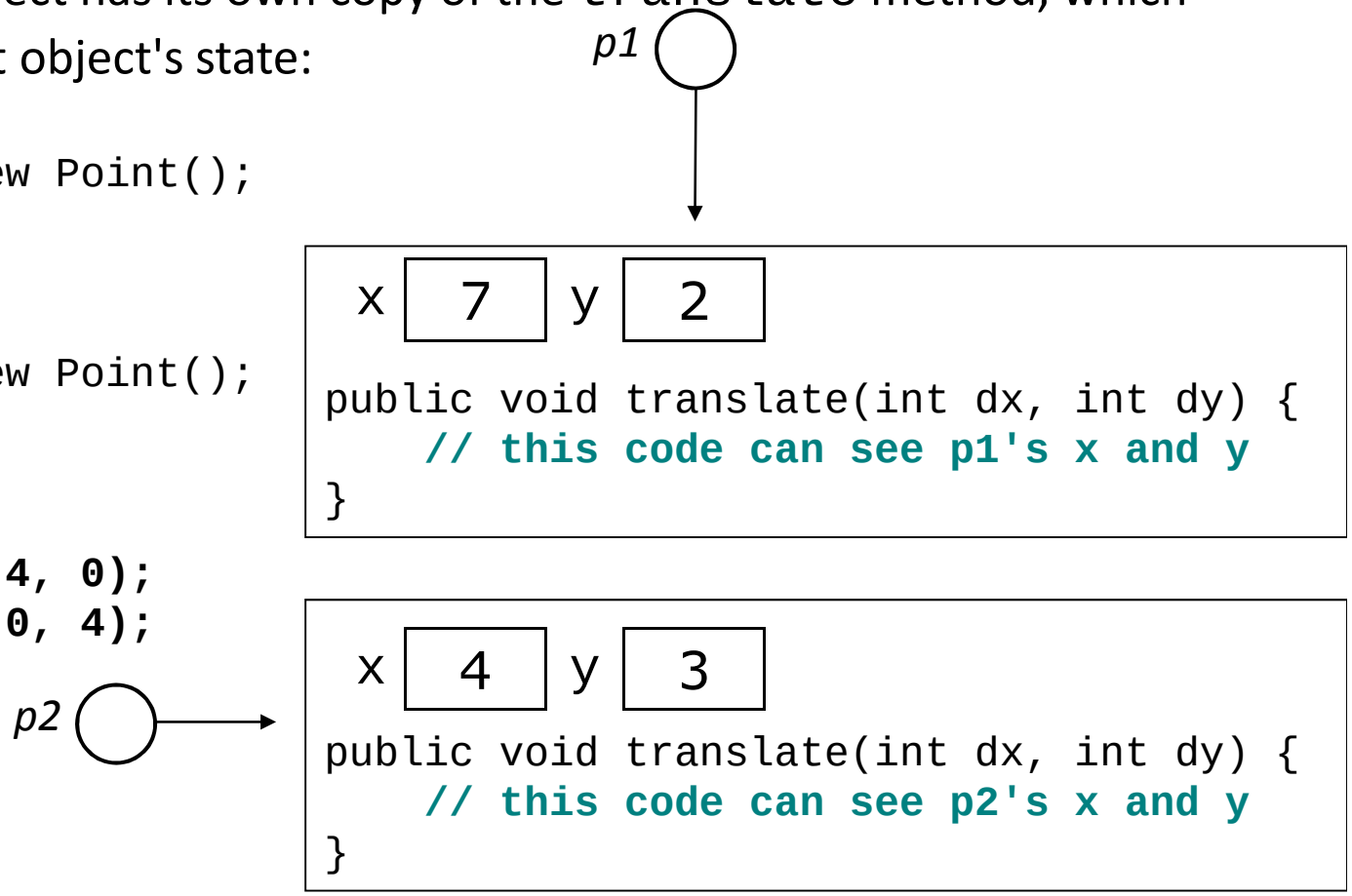
`Point` objects w/ method

- Each `Point` object has its own copy of the `translate` method, which operates on that 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(4, 0);
p2.translate(0, 4);
```



Kinds of methods

- Instance methods take advantage of an object's state.
  - Some methods allow clients to access/modify its state.

- **accessor**: A method that lets clients examine object state.
  - Example: A `distanceFromOrigin` method that tells how far a `Point` is away from (0, 0).
  - Accessors often have a non-`void` return type.

- **mutator**: A method that modifies an object's state.
  - Example: A `translate` method that shifts the position of a `Point` by a given amount.

Initializing objects

- Currently it takes 3 lines to create a `Point` and initialize it:

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

- We'd rather pass the fields' initial values as parameters:

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

  - We are able to this with most types of objects in Java.

Constructors

- **constructor**: Initializes the state of new objects.

```
public type(parameters) {
    statements;
}
```

  - runs when the client uses the new keyword
  - does not specify a return type;
    it implicitly returns the new object being created
  - If a class has no constructor, Java gives it a *default constructor* with no parameters that sets all fields to 0.

Common constructor bugs

- Accidentally writing a return type such as `void`:

```
public void Point(int initialX, int initialY) {
    x = initialX;
    y = initialY;
}
```

  - This is not a constructor at all, but a method!

- Storing into local variables instead of fields ("shadowing"):

```
public Point(int initialX, int initialY) {
    int x = initialX;
    int y = initialY;
}
```

  - This declares local variables with the same name as the fields, rather than storing values into the fields. The fields remain 0.