Data Structure Design II
Chris Piech
CS106A, Stanford University
Today in lecture
We have used many variable types
E.g. GRect
E.g. String
E.g. AudioSample
Today we learn how to define our own
We use Classes (written in new files) to defined new variable types
You must define three things

1. What **variables** does each instance store?

2. What **methods** can you call on an instance?

3. What happens when you make a **new** one?

*details on how to define these three things coming soon*
Classes are like blueprints

class: A template for a new type of variable.

A blueprint is a helpful analogy
What Does GRect Look Like?

public class GRect extends GObject {

GRect.java
public class GRect extends GObject {

    // Instance variables
    private double width = 0;
    private double height = 0;
    private double yc = 0;
    private double xc = 0;
    private boolean isFilled = false;
    private boolean isVisible = false;

    ...
public class GRect extends GObject {

    // Instance variables
    private double width = 0;
    private double height = 0;
    private double yc = 0;
    private double xc = 0;
    private boolean isFilled = false;
    private boolean isVisible = false;

    // Constructors
    public GRect(double width, double height) {
        this.width = width;
        this.height = height;
    }
    public GRect(double x, double y, double width, double height) {
        this.xc = x;
        this.yc = y;
        this.width = width;
        this.height = height;
    }

    ...
}
What Does GRect Look Like?

```java
// Public methods
public double getWidth() {
    return this.width;
}

public double getHeight() {
    return this.height;
}

public void setFilled(boolean newIsFilled) {
    this.isFilled = newIsFilled;
}

public void move(double dx, double dy) {
    this.xc += dx;
    this.yc += dy;
}

... and so on ...

GRect.java
```
What does a class do?
A class defines a new variable type
Bouncing Balls
A Ball Variable Type

The Ball class

1. What **variables** does each instance store?
   - Each ball has its own Goval (let's call it shape)
   - Each ball has its own dx
   - Each ball has its own dy

2. What **methods** can you call on an instance?
   - `heartbeat();`
   - `getShape();`

3. What happens when you make a **new** one?
   - Sets initial values for all the "instance" vars

*details on how to define these three things coming soon*
public class Ball {
    /* instance vars! */

    // each ball has a “shape"
    private GOval shape = null;

    // each ball has a dx
    private double dx = 0.0;

    // each ball has a dy
    private double dy = 0.0;

    ...

    1. Instance vars define what makes up a variable of type Ball
public class Ball {
    /* instance vars! */

    // each ball has a "shape"
    private GOval shape = null;

    // each ball has a dx
    private double dx = 0.0;

    // each ball has a dy
    private double dy = 0.0;

    // This defines what happens when you make a new ball
    public Ball(int screenWidth, int screenHeight) {
        RandomGenerator rg = RandomGenerator.getInstance();
        double x = rg.nextInt(screenWidth - BALL_SIZE);
        double y = rg.nextInt(screenHeight - BALL_SIZE);
        shape = new GOval(x, y, BALL_SIZE, BALL_SIZE);
        shape.setFilled(true);
        shape.setColor(Color.BLUE);
        dx = getRandomSpeed();
        dy = getRandomSpeed();
    }

    ...
public void heartbeat(int screenWidth, int screenHeight) {
    shape.move(dx, dy);
    reflectOffWalls(screenWidth, screenHeight);
}

public GOval getShape() {
    return shape;
}

...
public void heartbeat(int screenWidth, int screenHeight) {
    shape.move(dx, dy);
    reflectOffWalls(screenWidth, screenHeight);
}

public GOval getShape() {
    return shape;
}

private void reflectOffWalls(int sWidth, int sHeight) {
    if(shape.getY() < 0) {
        dy *= -1;
    }
    if(shape.getY() > sHeight - BALL_SIZE) {
        dy *= -1;
    }
    if(shape.getX() < 0) {
        dx *= -1;
    }
    if(shape.getX() > sWidth - BALL_SIZE) {
        dx *= -1;
    }
}
What does a class do?
A class defines a new variable type
You must define three things

1. What **variables** does each instance store?

2. What **methods** can you call on an instance?

3. What happens when you make a **new** one?
Wait... if each instance has a copy of each instance variable. How does Java know which one to use?
public class BouncingBalls extends GraphicsProgram {
    public void run() {
        // make a few new balls
        Ball a = new Ball(getWidth(), getHeight());
        Ball b = new Ball(getWidth(), getHeight());

        // call a method on one of the balls
        a.heartbeat(getWidth(), getHeight());
    }
}
public class BouncingBalls extends GraphicsProgram {

public void run() {
    // make a few new balls
    Ball a = new Ball(getWidth(), getHeight());
    Ball b = new Ball(getWidth(), getHeight());
    // call a method on one of the balls
    a.heartbeat(getWidth(), getHeight());
}

public void heartbeat(int screenWidth, int screenHeight) {
    shape.move(dx, dy);
    reflectOffWalls(screenWidth, screenHeight);
}

run

a

b

heartbeat

this

dsWidth  800

sHidht  600

dx = 1.0

dy = 1.5

800

600
```java
public class BouncingBalls extends GraphicsProgram {

    public void run() {
        // make a few new balls
        Ball a = new Ball(getWidth(), getHeight());
        Ball b = new Ball(getWidth(), getHeight());

        // call a method on one of the balls
        a.heartbeat(getWidth(), getHeight());
    }

    public void heartbeat(int screenWidth, int screenHeight) {
        shape.move(dx, dy);
        reflectOffWalls(screenWidth, screenHeight);
    }
}
```

- run
  - a
  - b

- heartbeat
  - this
  - sWidth: 800
  - sHeight: 600

- Variables:
  - dx = 1.0
  - dy = 1.5

- After heartbeat:
  - dx = -1.2
  - dy = -1.1
Tl;dr: Java knows which Ball you called heartbeat on
public class GRect extends GObject {

    // Instance variables
    private double width = 0;
    private double height = 0;
    private double yc = 0;
    private double xc = 0;
    private boolean isFilled = false;
    private boolean isVisible = false;

    // Constructors
    public GRect(double width, double height) {
        this.width = width;
        this.height = height;
    }
    public GRect(double x, double y, double width, double height) {
        this.xc = x;
        this.yc = y;
        this.width = width;
        this.height = height;
    }

    ...

    GRect.java
Hello from lecture

Dear [Name],

I hope this email finds you well.

As you know, CS106A is a huge class with many wonderful people in it. In lecture today we built a program to help you meet a few fellow students. Here are five random people in CS106A. You can (optionally) introduce yourself:
- Omar, omar@stanford.edu
- Micah, micah@stanford.edu
- Gianfranco, gianfranco@stanford.edu
- Noam, noam@stanford.edu
- Dylan, dylan@stanford.edu

All the best,
Chris

P.S. Today we covered 'classes' which introduces a whole new way of thinking about programs.
Wall of abstraction
Adding Privacy

```java
private boolean isLeftImgShown;
```

- **encapsulation**: Hiding implementation details of an object from its clients.
  - Encapsulation provides *abstraction*.
    - separates external view (behavior) from internal view (state)
  - Encapsulation protects the integrity of an object's data.

- A class's instance variables should be declared *private*.
  - No code outside the class can access or change it.
What does a class do?
A class defines a new variable type
You must define three things

1. What **variables** does each instance store?

2. What **methods** can you call on an instance?

3. What happens when you make a **new** one?
More Practice

See Days Until

CALENDAR

1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

Piech, CS106A, Stanford University