Classes

CS106A, Stanford University
Laura Cruz-Albrecht
What do we know about classes?
Classes define a new variable type
Classes are like blueprints

```
public class House {
    private int nRooms;
    private double height;

    public House(int nRooms, double height) {
        this.numRooms = numRooms;
        this.height = height;
    }
    ...
}
```

```
House myHouse = new House(2, 200);
House brahmsHouse = new House(5, 300);
```
Making a class - 3 ingredients

1. Define the variables each instance stores (think: properties)

2. Define the constructor used to make a new instance

3. Define the methods you can call on an instance (think: behaviors)
You’ve seen them before...
GRect square = new GRect(10, 10);

GRect.java

public class GRect {
    public GRect(double width, double height) {
        this.width = width;
        this.height = height;
    }
    ...
}

type our object (variable) It's an instance of the GRect class!
double x = square.getX()

Method defined in GRect class that we can call on our object
Unpacking GRect

```java
public class GRect {
    private double width;
    public GRect(double width, double height) {
        ...}
    ...
}
```
public class GRect {

3 Ingredients:
public class GRect {

    // 1. Instance variables
    private double width = 0;
    private double height = 0;
    private double yc = 0;
    private double xc = 0;
    private boolean filled = false;
    private boolean isVisible = false;

    GRect.java

    3 Ingredients:

    1. Define the variables each instance stores
public class GRect {

    // 1. 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;

    // 2. Constructor(s)
    public GRect(double width, double height) {
        this.width = width;
        this.height = height;
    }

    // 3. Ingredients:
    // 1. Define the variables each instance stores
    // 2. Define the constructor used to make a new instance
public class GRect {

    // 1. 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;

    // 2. Constructor(s)
    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;
    }
}

3 Ingredients:

1. Define the variables each instance stores

2. Define the constructor used to make a new instance
public class GRect {

    // 1. 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;

    // 2. Constructor(s)
    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;
    }

    // 3. 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;
    }
}

3 Ingredients:

1. Define the variables each instance stores
2. Define the constructor used to make a new instance
3. Define the methods you can call on an instance
Making our own classes
Back to Bouncing Ball...
Making a Ball variable type

1. Define the variables each instance stores (think: properties)
   - Each ball has its own GOval (let’s call it circle)
   - Each ball has its own dx
   - Each ball has its own dy

2. Define the constructor used to make a new instance
   Set initial values for all the instance vars

3. Define the methods you can call on an instance (think: behaviors)
   - heartbeat()
   - getGOval()
public class Ball {

    private static final int BALL_SIZE = 20;

    // 1: what variables make up a ball?
    private GOval circle; // each ball has a GOval shape
    private double dx; // each ball has a dx
    private double dy; // each ball has a dy

    public Ball() {
        // make the ball's circle
        this.circle = new GOval(0, 0, BALL_SIZE, BALL_SIZE);
        this.circle.setFilled(true);
        this.circle.setColor(Color.BLUE);

        // gets a random dx and a random dy
        this.dx = getRandomSpeed();
        this.dy = getRandomSpeed();
    }

    1. Instance variables define what makes up a variable of type Ball
public class Ball {

    private static final int BALL_SIZE = 20;

    // 1: what variables make up a ball?
    private G Oval circle;  // each ball has a GOval shape
    private double dx;     // each ball has a dx
    private double dy;     // each ball has a dy

    // 2. what happens when you make a new ball?
    public Ball() {
        // make the ball's circle
        this.circle = new GOval(0, 0, BALL_SIZE, BALL_SIZE);
        this.circle.setFilled(true);
        this.circle.setColor(Color.BLU E);

        // gets a random dx and a random dy
        this.dx = getRandomSpeed();
        this.dy = getRandomSpeed();
    }
}
// 3. what methods can you call on a ball?
public GOval getGOval() {
    return this.circle;
}

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

// private methods are allowed
private void reflectOffWalls(int screenWidth, int screenHeight) {
    if (this.circle.getY() < 0) {
        this.dy *= -1;
    }
    if (this.circle.getY() > screenHeight - BALL_SIZE) {
        this.dy *= -1;
    }
    if (this.circle.getX() < 0) {
        this.dx *= -1;
    }
    if (this.circle.getX() > screenWidth - BALL_SIZE) {
        this.dx *= -1;
    }
}

3. Public methods define what the “client” can call on instances
private void reflectOffWalls(int screenWidth, int screenHeight) {
    if (this.circle.getY() < 0) {
        this.dy *= -1;
    }
    if (this.circle.getY() > screenHeight - BALL_SIZE) {
        this.dy *= -1;
    }
    if (this.circle.getX() < 0) {
        this.dx *= -1;
    }
    if (this.circle.getX() > screenWidth - BALL_SIZE) {
        this.dx *= -1;
    }
}

private double getRandomSpeed() {
    RandomGenerator rg = RandomGenerator.getInstance();
    double speed = rg.nextDouble(1, 3);
    if (rg.nextBoolean()) {
        speed *= -1;
    }
    return speed;
}
But if each Ball instance has a copy of each instance variable...

... how does Java know which one to use?
* all class methods and constructors have access to a **this** reference
public class BouncingBall extends GraphicsProgram {
    public void run() {
        // make a few new bouncing balls
        Ball a = new Ball();
        Ball b = new Ball();

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

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

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

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

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

public void heartbeat(int sWidth, int sHeight) {
    this.circle.move();
    reflectOffWalls(sWidth, sHeight);
}

Stack frames
run()

heartbeat()

heap

circle

42

52
dx = 1.0
dy = 1.5

circle

42
dx = -1.2
dy = -1.1

memory

stack

run()

heartbeat()
public class BouncingBall extends GraphicsProgram {
    public void run() {
        // make a few new bouncing balls
        Ball a = new Ball();
        Ball b = new Ball();

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

public void heartbeat(int sWidth, int sHeight) {
    this.circle.move();
    reflectOffWalls(sWidth, sHeight);
}

heartbeat() was called on ball a ⇒ So, this refers to a

Stack frames

run()

| a | 42 |
| b | 52 |

heartbeat()

| this |
| sWidth | 800 |
| sHeight | 600 |

heap

circle

dx = 1.0
dy = 1.5

circle

dx = -1.2
dy = -1.1
public class BouncingBall extends GraphicsProgram {
    public void run() {
        // make a few new bouncing balls
        Ball a = new Ball();
        Ball b = new Ball();

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

public void heartbeat(int sWidth, int sHeight) {
    this.circle.move();
    reflectOffWalls(sWidth, sHeight);
}

Heartbeat() was called on ball a ⇒ So, this refers to a
public class BouncingBall extends GraphicsProgram {
    public void run() {
        // make a few new bouncing balls
        Ball a = new Ball();
        Ball b = new Ball();

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

public void heartbeat(int sWidth, int sHeight) {
    this.circle.move();
    reflectOffWalls(sWidth, sHeight);
}

heartbeat() was called on ball a
⇒ So, this refers to a
Java knows which instance you called a method on
One more note

ArrayList<Ball> balls

Index 0  Index 1  Index 2
42  52  62

circle
dx = -1.2
dy = -1.1

42  52  62

circle
dx = 2.0
dy = 1.5

circle
dx = 1.8
dy = -2.2
`ArrayList<Ball> balls

Ball newBall = new Ball()`
balls.append(newBall)
balls.append(newBall)
Let’s build something bigger
Sending emails...
To: [email protected]@stanford.edu
Subject: Greetings from Lecture
Text:
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:

Jordan, [email protected]@stanford.edu
Catherine, [email protected]@stanford.edu
Raushun, [email protected]@stanford.edu
Lora, [email protected]@stanford.edu
Monica, [email protected]@stanford.edu

All the best,
Laura (and Chris :))

P.S. Today we covered 'classes' which introduces a whole new way of thinking about programs
More practice
What’s in a Guinea Pig?

Guinea Pigs have properties and behaviors

Properties → ...

Behaviors → ...

How do we model this with classes?

Properties → Instance variables

Behaviors → Methods
Making a Guinea Pig variable type

1. Define the **variables** each instance stores (think: properties)
   Name, color, age, likes to squeak

2. Define the **constructor** used to make a **new** instance
   Initialize our instance variables

3. Define the **methods** you can call on an instance (think: behaviors)
   Getters & setters (to access/modify properties), squeak()
public class GuineaPig {

    // 1. Instance variables
    private String name;
    private String color;
    private int age;
    private boolean likesToSqueak;
}
public class GuineaPig {

    // 1. Instance variables
    private String name;
    private String color;
    private int age;
    private boolean likesToSqueak;

    // 2. Constructor
    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        this.color = color;
        this.age = age;
        this.likesToSqueak = likesToSqueak;
    }
}
public class GuineaPig {

    // 1. Instance variables
    private String name;
    private String color;
    private int age;
    private boolean likesToSqueak;

    // 2. Constructor
    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        this.color = color;
        this.age = age;
        this.likesToSqueak = likesToSqueak;
    }

    // 3. Methods
    public String getName() { // Getters & setters
        return this.name;
    }

    public int getAge() {
        return this.age;
    }

    public void setAge(int newAge) {
        this.age = newAge;
    }
}
public class GuineaPig {

    // 1. Instance variables
    private String name;
    private String color;
    private int age;
    private boolean likesToSqueak;

    // 2. Constructor
    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        this.color = color;
        this.age = age;
        this.likesToSqueak = likesToSqueak;
    }

    // 3. Methods
    public String getName() { // Getters & setters
        return this.name;
    }

    public int getAge() {
        return this.age;
    }

    public void setAge(int newAge) {
        this.age = newAge;
    }

    public String squeak() { // behaviors
        String squeakStr = "Squeak."
        if (this.likesToSqueak) {
            squeakStr += " Squeak, squeak!";
        }
        return "I’m " + this.name + ". " + squeakStr;
    }
}
public class GuineaPig {

  // 1. Instance variables
  private String name;
  private String color;
  private int age;
  private boolean likesToSqueak;

  // 2. Constructor
  public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
    this.name = name;
    this.color = color;
    this.age = age;
    this.likesToSqueak = likesToSqueak;
  }

  // 3. Methods
  public String getName() { // Getters & setters
    return this.name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int newAge) {
    this.age = newAge;
  }

  public String squeak() { // behaviors
    String squeakStr = "Squeak."
    if (this.likesToSqueak) {
      squeakStr += " Squeak, squeak!";
    }
    return "I'm " + this.name + ". " + squeakStr;
  }

  public String toString() { // to String
    return "Guinea Pig: " + this.name;
  }
}
We can now make GuineaPigs!

GuineaPig.java

```java
public class GuineaPig {
    private String name;
    ...

    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        ...
    }

    public String squeak() {
        String squeakStr = " Squeak."
        if (this.likesToSqueak) {
            squeakStr += " Squeak, squeak!"
        }
        return "I'm " + this.name + ". " + squeakStr;
    }

    public String toString() {
        return "Guinea Pig: " + this.name;
    }
}
```

MyPets.java

```java
Public class MyPets extends ConsoleProgram {
    public void run() {
        GuineaPig walnut = new GuineaPig("Walnut", "brown", 3, false);
        GuineaPig chestnut = new GuineaPig("Chestnut", "beige", 2, true);
    }
}
```

console
We can now make GuineaPigs!

GuineaPig.java

```java
public class GuineaPig {
    private String name;
    ...

    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        ...
    }

    public String squeak() {
        String squeakStr = " Squeak."
        if (this.likesToSqueak) {
            squeakStr += " Squeak, squeak!";
        }
        return "I'm " + this.name + ". " + squeakStr;
    }

    public String toString() {
        return "Guinea Pig: " + this.name;
    }
}
```

MyPets.java

```java
Public class MyPets extends ConsoleProgram {
    public void run() {
        GuineaPig walnut = new GuineaPig("Walnut", "brown", 3, false);
        GuineaPig chestnut = new GuineaPig("Chestnut", "beige", 2, true);

        println(walnut); // toString

        walnut.setAge(walnut.getAge() + 1);
        println(walnut.getName() + "'s age: " + walnut.getAge());
        println(chestnut.squeak());
    }
}
```

console

Guinea Pig: Walnut
We can now make GuineaPigs!

GuineaPig.java

```java
public class GuineaPig {
    private String name;
    ...

    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
    }

    public String squeak() {
        String squeakStr = " Squeak."
        if (this.likesToSqueak) {
            squeakStr += " Squeak, squeak!";
        }
        return "I'm " + this.name + ". " + squeakStr;
    }

    public String toString() {
        return "Guinea Pig: " + this.name;
    }
}
```

MyPets.java

```java
Public class MyPets extends ConsoleProgram {
    public void run() {
        GuineaPig walnut = new GuineaPig("Walnut", "brown", 3, false);
        GuineaPig chestnut = new GuineaPig("Chestnut", "beige", 2, true);

        println(walnut); // toString
        walnut.setAge(walnut.getAge() + 1);
        println(walnut.getName() + ", ", walnut.getAge());
    }
}
```

```
Guinea Pig: Walnut
Walnut’s age: 4
```

console
We can now make GuineaPigs!

GuineaPig.java

```java
public class GuineaPig {
    private String name;
    ...

    public GuineaPig(String name, String color, int age, boolean likesToSqueak) {
        this.name = name;
        ...
    }

    public String squeak() {
        String squeakStr = " Squeak."
        if (this.likesToSqueak) {
            squeakStr += " Squeak, squeak!";
        }
        return "I’m " + this.name + ". " + squeakStr;
    }

    public String toString() {
        return "Guinea Pig: " + this.name;
    }

}```

MyPets.java

```java
public class MyPets extends ConsoleProgram {
    public void run() {
        GuineaPig walnut = new GuineaPig("Walnut", "brown", 3, false);
        GuineaPig chestnut = new GuineaPig("Chestnut", "beige", 2, true);

        println(walnut); // toString
        walnut.setAge(walnut.getAge() + 1);
        println(walnut.getName() + "’s age: " + walnut.getAge());
        println(chestnut.squeak());
    }
}
```

console

Guinea Pig: Walnut
Walnut’s age: 4
I’m Chestnut. Squeak. Squeak, squeak!