Java Puzzle Ball
Nick Ristuccia

Lesson 1-3
Editing Java Code
Java Puzzle Ball Code is Complex

```java
theta = Math.toRadians(180-theta);
double r = image.getHeight();
double x = r*Math.sin(theta) -(image.getWidth()/2)*Math.cos(theta) +pivotX;
double y = r*Math.cos(theta) +(image.getWidth()/2)*Math.sin(theta) +pivotY;
p1.setLocation(x,y);

x = r*Math.sin(theta) +(image.getWidth()/2)*Math.cos(theta) +pivotX;
y = r*Math.cos(theta) -(image.getWidth()/2)*Math.sin(theta) +pivotY;
p2.setLocation(x,y);

r = 0;
x = r*Math.sin(theta) +pivotX;
y = r*Math.cos(theta) +pivotY;
p3.setLocation(x,y);

walls.get(0).setLine(p1,p2);
walls.get(1).setLine(p2,p3);
walls.get(2).setLine(p3,p1);
```
if(javafxapplication01.Ball.getSingletonBall().getIsBladeBall() == true && !isDestroyed){
    javafxapplication01.Ball.getSingletonBall().setBladeToBall();
    bumper.destroy();
}
if(!isDestroyed){
    return new GameObjectAction(true, new Behavior() {
        @Override
        public GameStatus step(Ball ball, Game game) {
            Level.sfxEngine.addSfx(new FireworksSFX(Level.sfxlayer,
                Level.game.getBall().getLocation(), 25, 7, 15,
                Level.game.getBall().getDirection()));
            bumper.bumpedByBall();
            Point2D destination = ball.calculateDestination();
            ball.setLocation(destination);
            ball.setBehavior(null);
            return GameStatus.RUNNING;
        }
    });
}
How will we Handle Complex Code?

• The previous two slides show just a portion of how the Triangle Wall code is implemented.

• It's complex.
  – In fact, it's too complex for this course.

• Solution: Abstract complex code
  – The Triangle Wall Icon ( ) represents this implementation in-game.
  – The syntax `TriangleWall()` also represents this implementation in examples.
This method is called `methodB()`.

Its implementation exists between two curly braces `{ }`.

The complex implementation is represented by `triangleWall()`.
You've Dictated how Methods should be Implemented...

```java
public class BlueBumper {
    private Color color = Color.BLUE;
    private Shape shape = Shape.RECT;
    private int xPosition;
    private int yPosition;

    public void methodA()
    {
        simpleWall();
    }

    public void methodB()
    {
        triangleWall();
    }

    public void methodC()
    {
    }

    public void methodD()
    {
    }
}
```
And Designed Classes to Accomplish your Goals

```java
public class BlueBumper {
    private Color color = Color.BLUE;
    private Shape shape = Shape.RECT;
    private int xPosition;
    private int yPosition;

    ...

    public void methodA(){
    }

    public void methodB(){
        fan();
    }

    public void methodC(){
    }

    public void methodD(){
        rotationWall();
    }
}
```
You've Come Far

• Think about what you now understand better:
  – Objects and Instances
  – Classes
  – Fields
  – Methods
  – Implementations
  – Java syntax

• Think about what problems you now have experience wrestling:
  – Planning and designing classes
  – Distributing methods wisely between classes
Your Hacking will be Sophisticated!

• Those are impressive and complex computer science topics!
• Now you're ready to apply this to editing your own Java code.
• You'll wisely edit code, not just hacking or guessing, to discover a solution
  – Because you have the conceptual understanding not to be helpless when faced with would-be walls of mystery syntax.
• This is how I learned programming at game studios:
  – Play with existing code to get a desired effect.
  – It would have gone faster if I understood what I was looking at.
  – You have advantages. Your hacking is "guided hacking" on a conceptual foundation.
Lab 1: Write a `CheckingAccount` class

- The Lab Instructions are available on the Lesson 1 page of the MOOC.
- As you work, consider...
  - What properties and behaviors are found in a checking account?
  - How can these be expressed through fields and methods?
- The remaining part of this lesson will give you tips.
The `SavingsAccount` class

```java
public class SavingsAccount {

// Fields
private String accountType;
private String accountOwner;
private double balance;
private double interestRate;

// Methods
public void printDetails(){
    ...
}
public void earnInterest(){
    ...
}
public void deposit(double x){
    ...
}
}
```

**Properties:**
- Account Type
- Account Owner
- Balance
- Interest Rate

**Behaviors:**
- Print Details
- Earn Interest
- Deposit
- Withdraw
The CheckingAccount class

```java
public class CheckingAccount {
    // Fields

    // Methods

    • Properties:
    – Account Type
    – Account Owner
    – Balance
    – Interest Rate

    • Behaviors:
    – Print Details
    – Earn Interest
    – Deposit
    – Withdraw
```
Study **SavingsAccount** to Build **CheckingAccount**

```java
public class SavingsAccount {
    //Fields
    private String accountType;
    private String accountOwner;
    private double balance;
    private double interestRate;

    //Methods
    public void printDetails()
    {
        ...
    }
    public void earnInterest()
    {
        ...
    }
    public void deposit(double x)
    {
        ...
    }
}
```

```java
public class CheckingAccount {
    //Fields

    //Methods
    public void printDetails()
    {
        ...
    }
}
```
public class TestClass {
    public static void main(String[] args) {

        // Create new instance
        SavingsAccount savings1 = new SavingsAccount();

        // Call methods on instance
        savings1.printDetails();
        savings1.deposit(5000);
        savings1.withdraw(100);
        savings1.earnInterest();

        // Create new instance
        CheckingAccount checking1 = new CheckingAccount();

        // Call methods on instance
    }
}
Lots More to Learn...

• What do `public` and `private` mean?
• What do `String`, `double`, and `void` mean?

• You won't need to know these keywords for this course.
  – But if you're curious, Oracle as other courses where you can learn more.