

# Introduction to Java

Michelle Brush  
Senior Software Architect  
Cerner Corporation



# Class Structure

- The Why and What of Java
- Basic Logic
- Object-Oriented Programming
- **When Things Go Wrong**
- Common Java APIs
- JUnit
- Generics
- Putting It All Together I & II



You will write bugs.

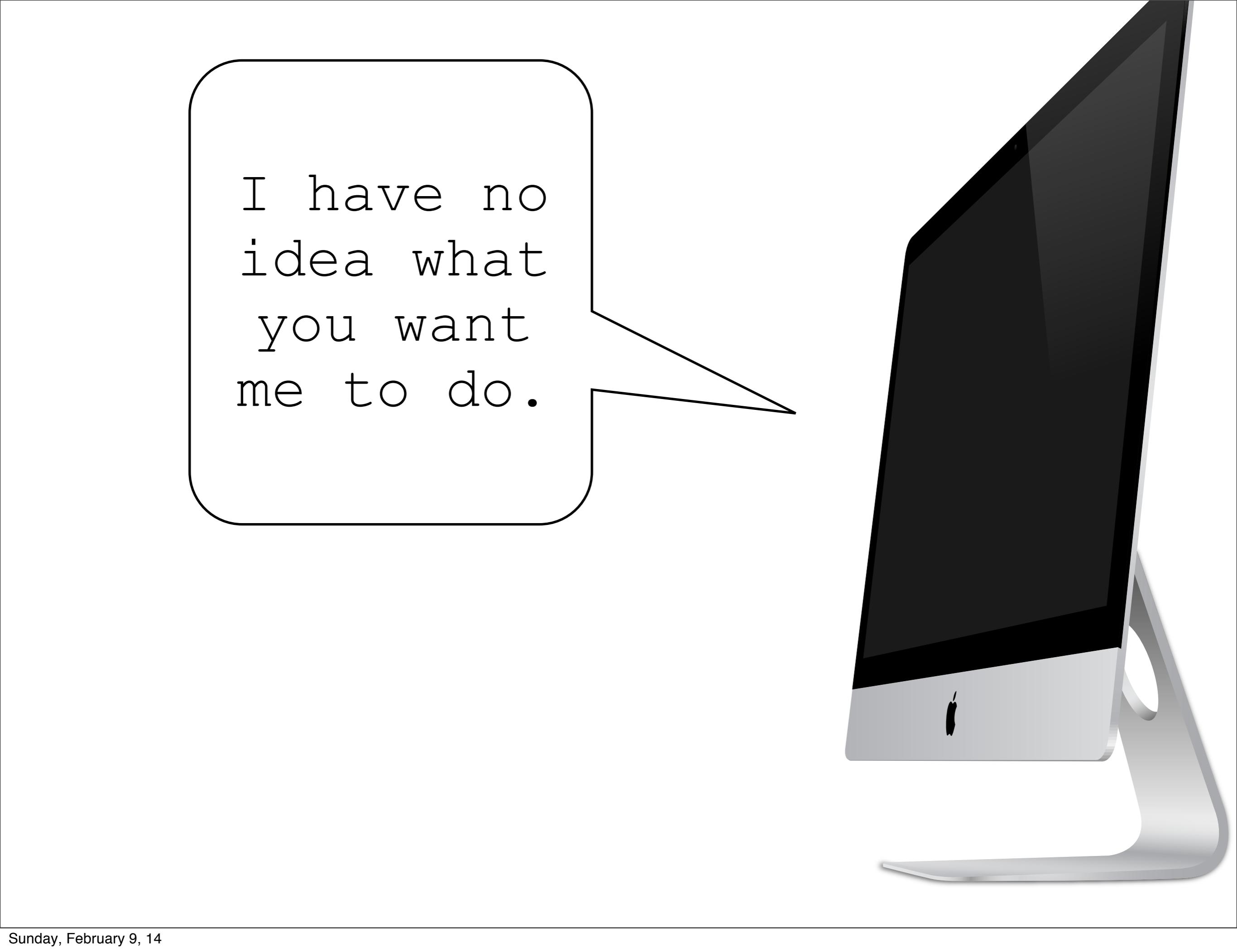
# Things that can go wrong....

- Compiler errors
- Runtime errors (bugs)
- Exceptions
- Input errors

# Compiler Errors

Programming languages have **syntax** that you must follow.

Compiler errors happen when your code doesn't follow the **syntax**.



I have no  
idea what  
you want  
me to do.

# Red Squiggles

The screenshot shows a Java code editor within an IDE. The project navigation bar at the top indicates the file path: Menu > src > com > girldevelopit > kc > menu > ConsolePrinter.java. The left sidebar displays the project structure under the 'Menu' package, including sub-packages like .idea, src, and com.girldevelopit.kc.menu, which contains classes like ConsolePrinter, DateRange, Food, HourRange, Item, ItemType, Menu, MenuPrinter, MenuRange, MenuSystem, and TemporalMenu. Below these are files Menu.iml and External Libraries. The main code editor window shows the 'ConsolePrinter.java' file. Several lines of code have red squiggly underlines underneath them, indicating syntax errors or warnings. The code itself is as follows:

```
package com.girldevelopit.kc.menu;

import java.util.Collections;
import java.util.Comparator;
import java.util.List;

/**
 * Created by michelle on 2/8/14.
 */
public class ConsolePrinter implements MenuPrinter {

    public void print(Menu menu) {
        System.out.println(menu.getName());

        List<Item> menuItems = menu.getItems();
        Collections.sort(menuItems, (Comparator) (o1, o2) -> {
            return o1.getType().compareTo(o2.getType());
        });

        ItemType type = null;
        for(Item item : menuItems) {
            if(type != item.getType()){
                System.out.println();
                System.out.println(item.getType().name());
                System.out.println();
            }

            String formattedPrice = formatPrice(item.getPrice());
            System.out.println(item.getName() + '\t' + formattedPrice);
            System.out.println(item.getDescription());
        }
    }
}
```

# Messages

Messages Make

The screenshot shows a 'Messages' panel from a Java development environment. On the left is a toolbar with icons for back, forward, search, and help. The main area displays the following log entries:

- Information: Using javac 1.7.0\_45 to compile java sources
- Information: java: Errors occurred while compiling module 'Menu'
- Information: Compilation completed with 1 error and 0 warnings in 3 sec
- Information: 1 error
- Information: 0 warnings

A detailed view of the error is expanded for the file `/Users/michelle/Menu/src/com/girldevelopit/kc/menu/ConsolePrinter.java`:

- Error:(26, 38) java: ')' expected

# How to Resolve the Error

- Read the error in the message view.
- Read the line of code it references.
- Check your assumptions.
- All else fails, Google it!

# Remember...

Computers are 3 year olds that are really good at math and have huge memories.

- They are obstinate.
- They have a poor vocabulary.
- They are never wrong.

# Bugs

1726

9/9

0800 arctan started  
1000 " stopped - arctan ✓  
13' UC (032) MP - MC  
(033) PRO 2  
cosine

{ 1.2700 9.037 847 025  
9.037 846 795 conduct  
~~1.982147000~~  
~~2.130476415~~ 4.615925059(-2)  
2.130676415

Relays 6-2 in 033 failed special speed test  
in relay " 10.000 test.

Relay  
2145

Relay 3376

1100 Started Cosine Tape (Sine check)  
1525 Started Multi Adder Test.

1545



Relay #70 Panel F  
(moth) in relay.

1630 arctangent started.  
1700 closed down.

First actual case of bug being found.

# Bugs

**Root cause:**

The code isn't doing what you think it is.

**Solution:**

Figure out what the code is really doing.

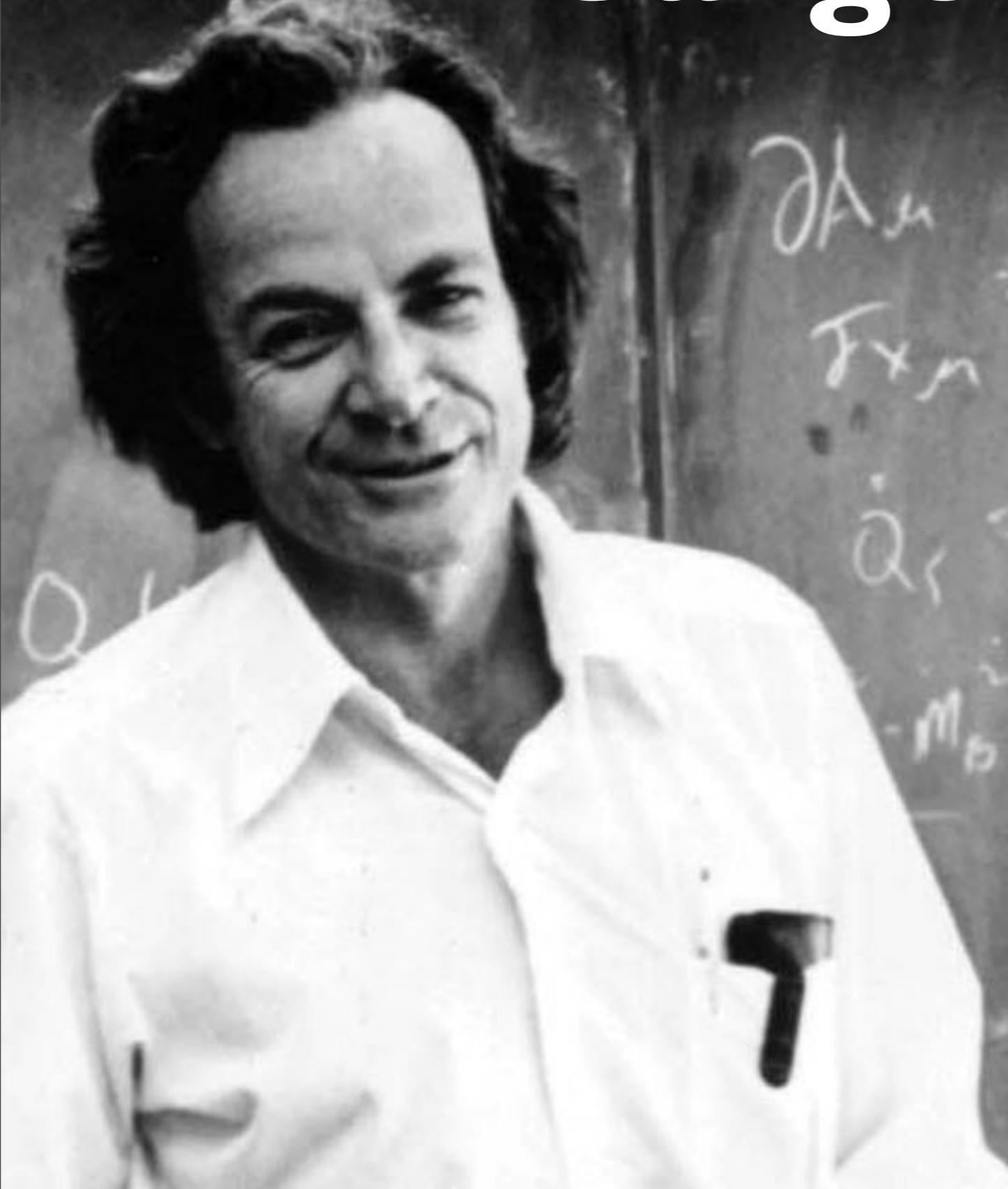
# Scientific Method



1. Characterize the problem.
2. Form a hypothesis.
3. Design an experiment.
4. Perform the Experiment.
5. Evaluate the result.

**Don't just throw code  
at the problem...**

# Cargo Cult



UXR  
wirkt mit Personen in  
Deutschland (y & p)  
wird mit Kiel wo 2

Wendet Metamaterial  
für akt. GHz-wellen

JMA HEGAU (KAUF  
MAN  
901-591-850

# Scientific Method



- 1. Characterize the problem.**
2. Form a hypothesis.
3. Design an experiment.
4. Perform the Experiment.
5. Evaluate the result.

# Characterize the Problem



Date:

In the box, draw & write as many words as you can to tell about your observation

# What **observations** can I make about the code?

Don't forget to use your five senses!

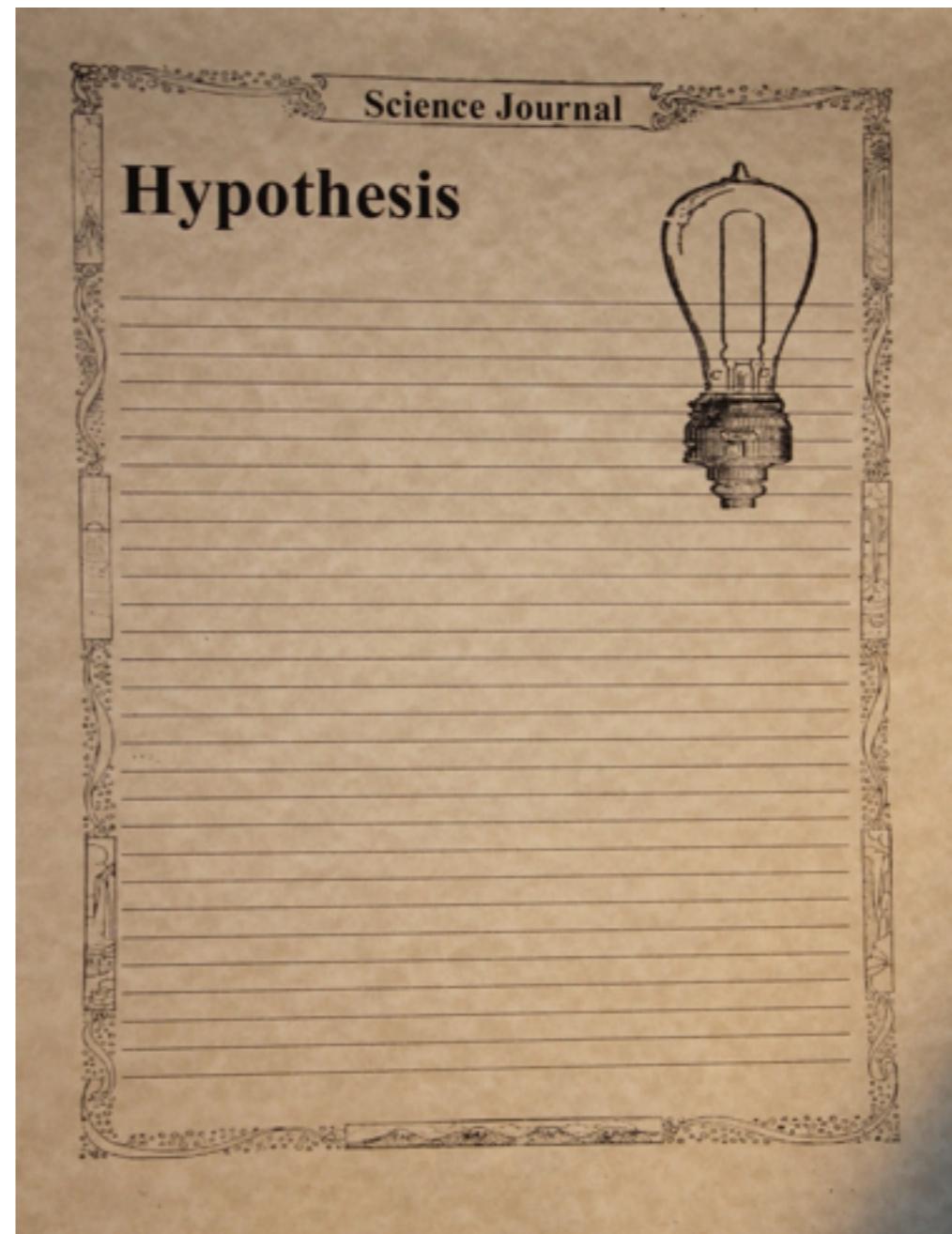
# Scientific Method



1. Characterize the problem.
- 2. Form a hypothesis.**
3. Design an experiment.
4. Perform the Experiment.
5. Evaluate the result.

# Form a Hypothesis

What is a reasonable **guess** as to the source of the issue?



# Scientific Method



1. Characterize the problem.
2. Form a hypothesis.
- 3. Design an experiment.**
4. Perform the Experiment.
5. Evaluate the result.

# Design an Experiment

What can I do  
to **test** my  
hypothesis?

The template is titled "Science EXPERIMENT". It includes fields for "Name", "Date", and "Title of Experiment". The main sections are:

- MATERIALS:** A list of materials used in the experiment.
- A QUICK SUMMARY OF WHAT YOU'RE TESTING OUT:** A summary of the hypothesis.
- Procedure:** A detailed description of the steps taken during the experiment.
- Hypothesis:** A dotted box for writing the hypothesis.
- Observations:** A large box for recording observations.
- WY DID IT HAPPEN?** A box for explaining the results.

# Design the Experiment

- Test
- Debug
- Eliminate Possibilities

# Scientific Method



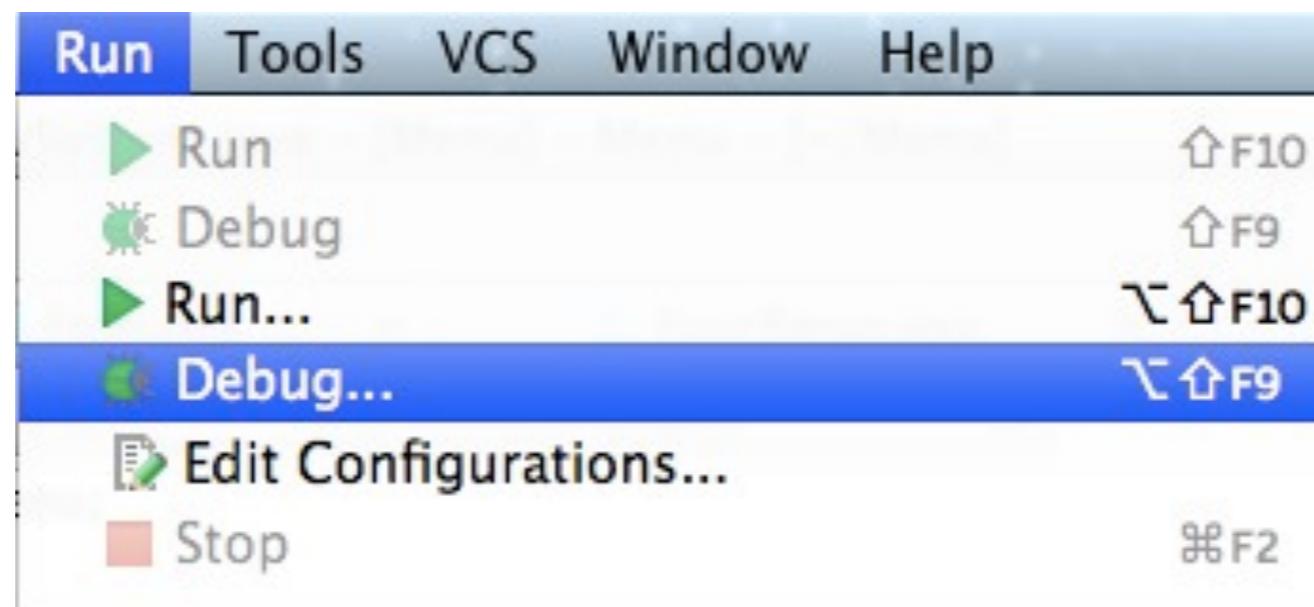
1. Characterize the problem.
2. Form a hypothesis.
3. Design an experiment.
- 4. Perform the Experiment.**
5. Evaluate the result.

# Test

- Try different inputs.
- Exercise different parts of the program.

Poke at the program.

# Debugger



# Debugging Terms

- Breakpoint
- Step over
- Step into
- Step out of



# Scientific Method



1. Characterize the problem.
2. Form a hypothesis.
3. Design an experiment.
4. Perform the Experiment.
- 5. Evaluate the result.**

# Evaluate the Result

- Did what I expected to happen, happen?
- Did I see something that surprised me?
- Is the cause obvious?

*If not, repeat the process.*

# Exceptions

Exceptions are objects.

They are “**thrown**” when something potentially bad happens.

You can “**catch**” them.

# Throwing

```
throw new RuntimeException("Error!");
```

# Catching

```
try {  
    //some operation  
} catch (Runtimeexception e) {  
    //handle it  
}
```

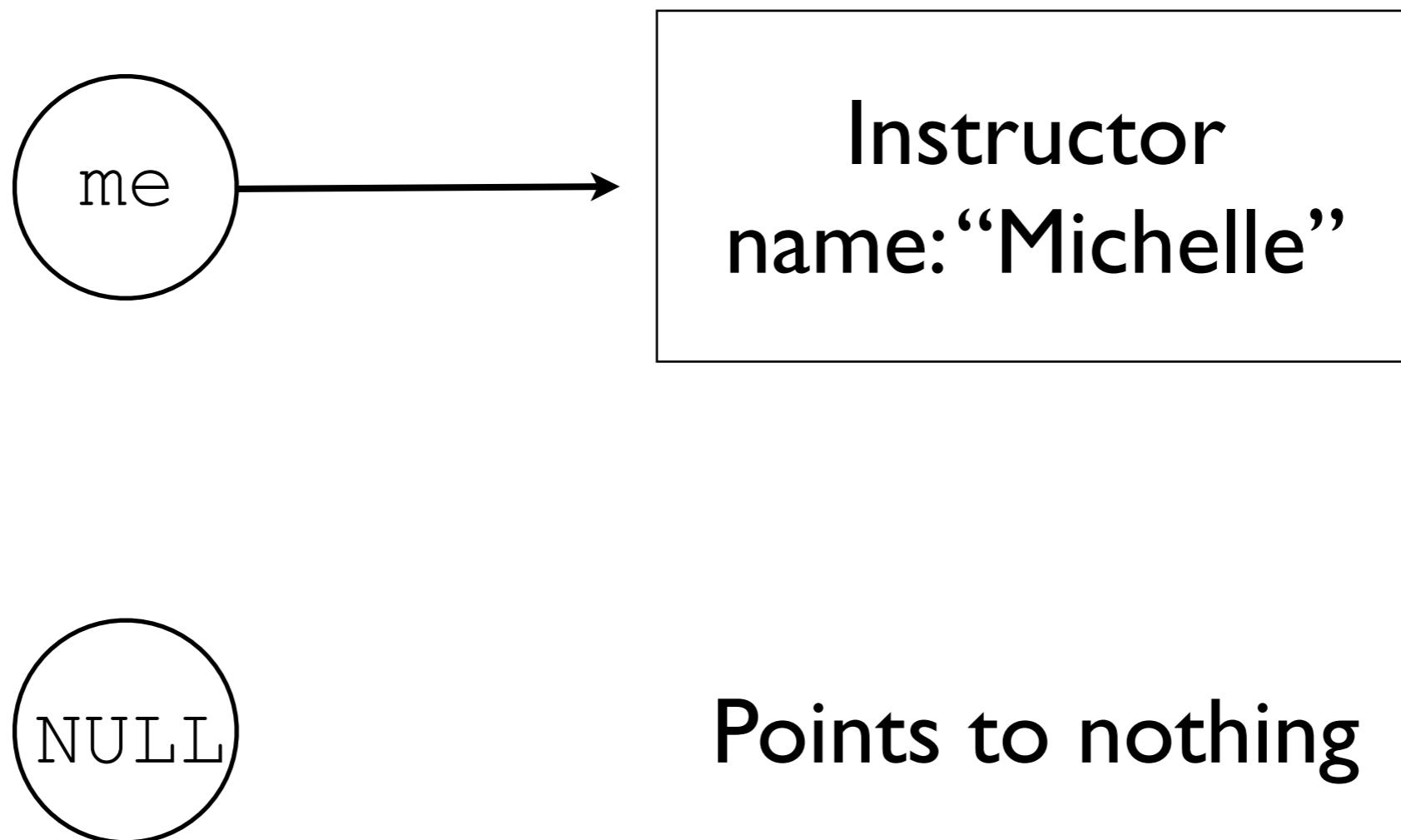
# Why?

- Your function gets bad parameters
- The computer ran out of memory
- A file couldn't be opened
- Something didn't exist
- Something isn't configured properly

# Most Common

An object is null.

# What is null?



# Null

Null is a reference to  
nothing.

# Where do null objects come from?

```
//We explicitly create them.  
Object obj = null;
```

# Where do null objects come from?

```
//We fail to initialize a member.  
public class Food {  
    private String name;  
    private String description;  
    public Food(String name) {  
        this.name = name;  
    }  
}
```

# Where do null objects come from?

```
//We get them from a function.  
Object obj = getObject();
```

# What happens if we use null?

```
Object obj = null;  
  
//Throws an exception  
obj.equals("happy");
```

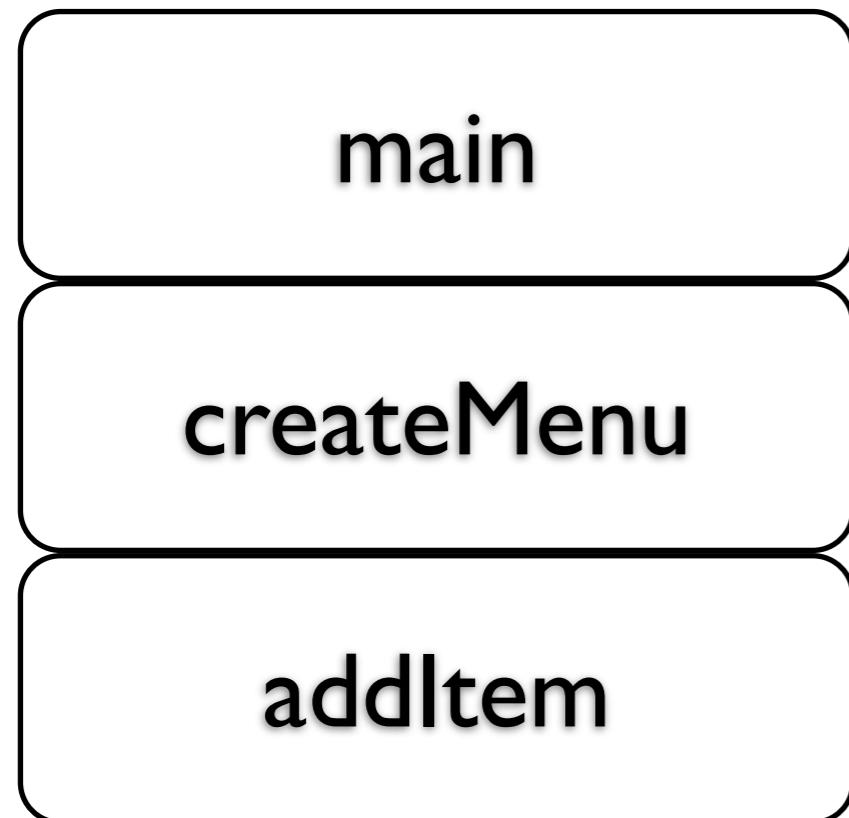
# NullPointerException

[http://docs.oracle.com/javase/7/docs/api/java/lang/  
NullPointerException.html](http://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html)

# RuntimeException

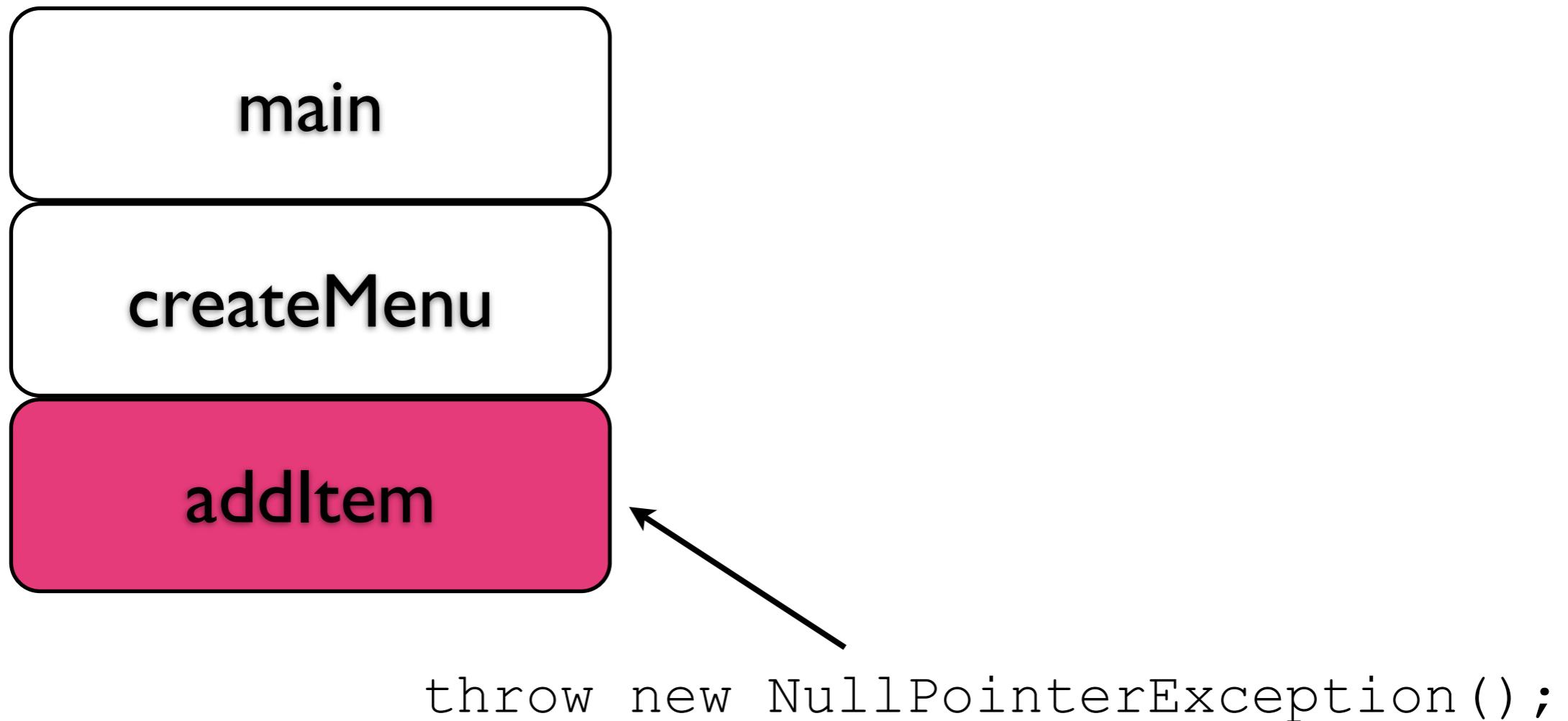
- You can catch them.
- If you don't, your program will terminate.

# Exceptions

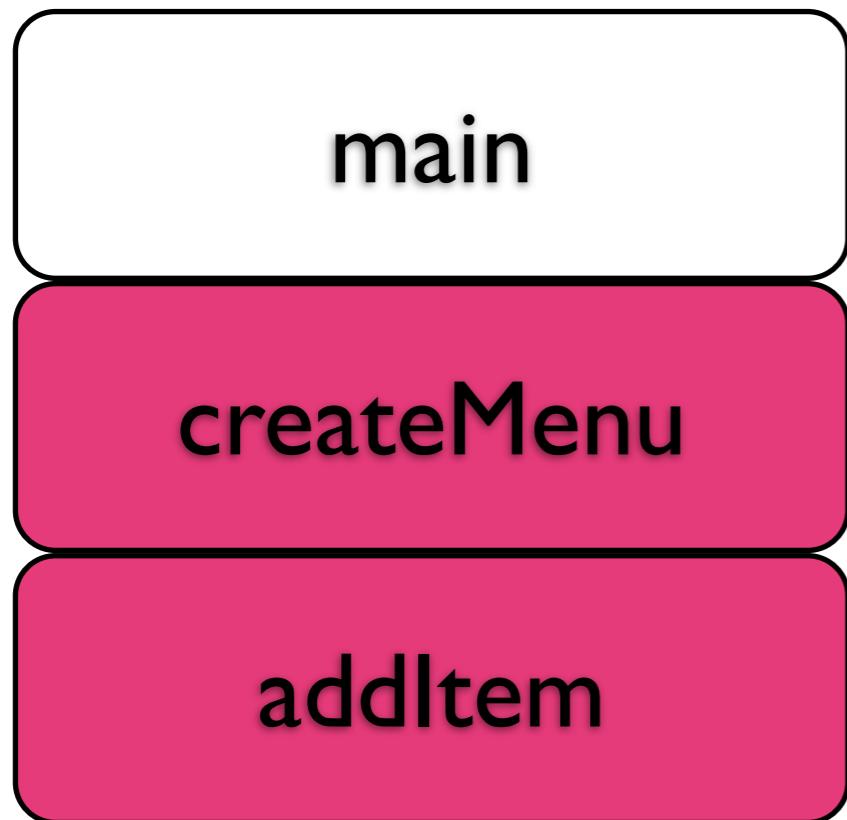


When an exception is thrown, the program immediately **jumps out** of the current function.

# Exceptions

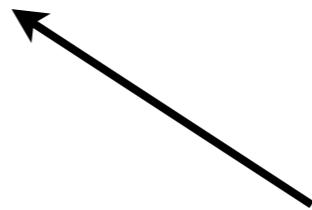
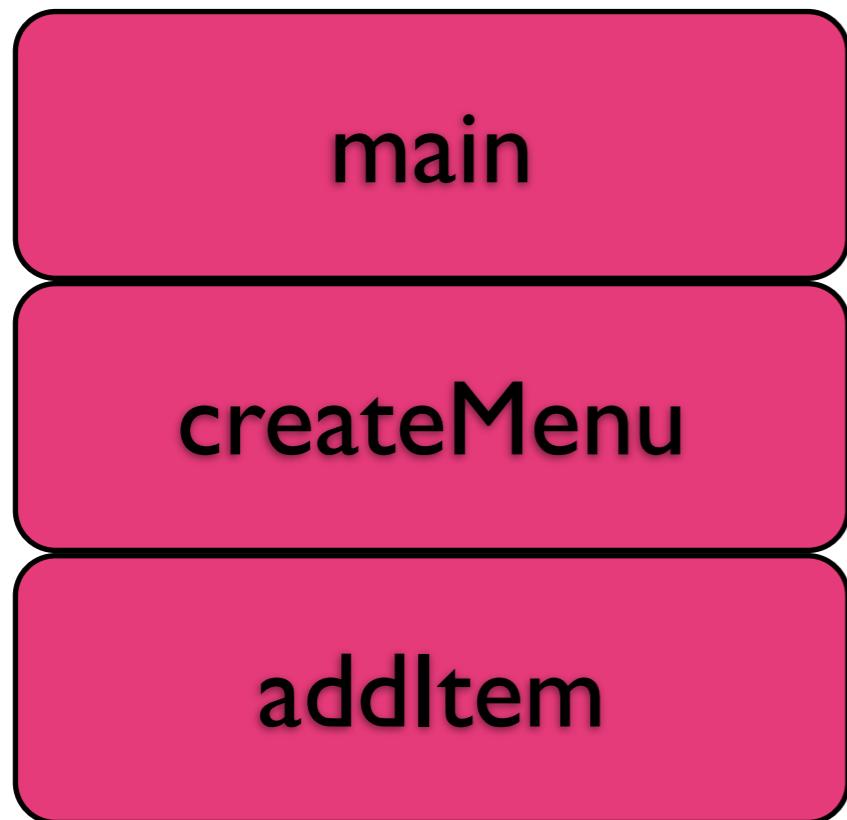


# Exceptions



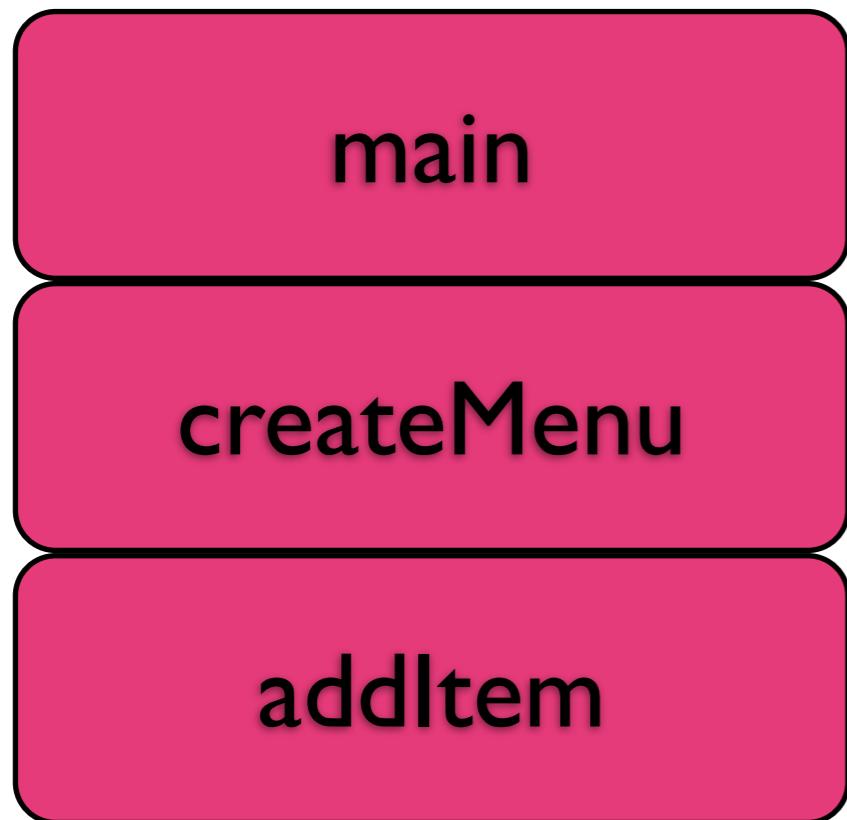
```
throw new NullPointerException();
```

# Exceptions

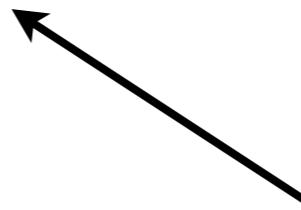


```
throw new NullPointerException();
```

# Exceptions

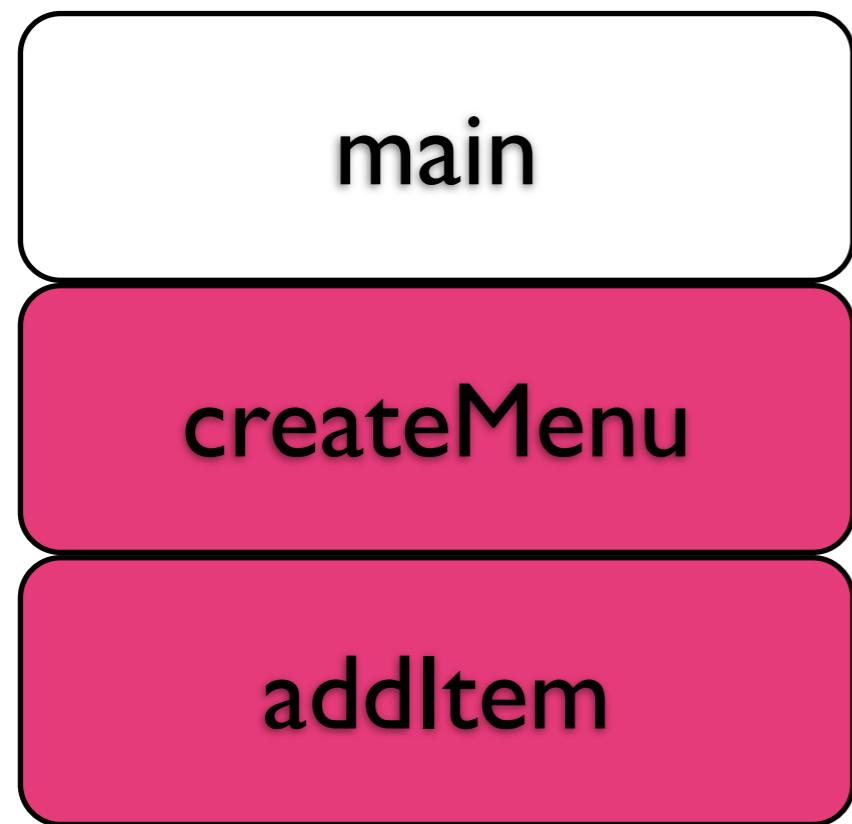


Program terminates.



```
throw new NullPointerException();
```

# Exceptions



```
try {  
    menu.addItem(...)  
} catch (NullPointerException e) {  
    //handling code  
}  
  
throw new NullPointerException();
```

# CheckedExceptions

- Compiler error if you don't recognize it.
  - try/catch
  - throws

# Proper Exceptions

- Only used for unexpected events
- Checked exceptions should be

# Proper Exceptions

- Only used for unexpected events
- Checked exceptions should be

# Creating Your Own

**Checked**

Exception class

**Unchecked**

RuntimeException class

<http://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html>

# Define reasonable...



# Creating Your Own

```
class MenuException extends RuntimeException {  
    public MenuException(String message) {  
        super(message);  
    }  
}
```

A close-up of two large, blue, cartoon-style eyes with white pupils and black pupils. The eyes are looking directly at the viewer. They are set against a green background that has a yellow diagonal stripe.

Fix my code.

# Download

<https://github.com/gdikc/IntroductionToJava>

Menu under samples