

# 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

# Things that can go wrong....

- Compiler errors
- Runtime errors (bugs)
- Exceptions

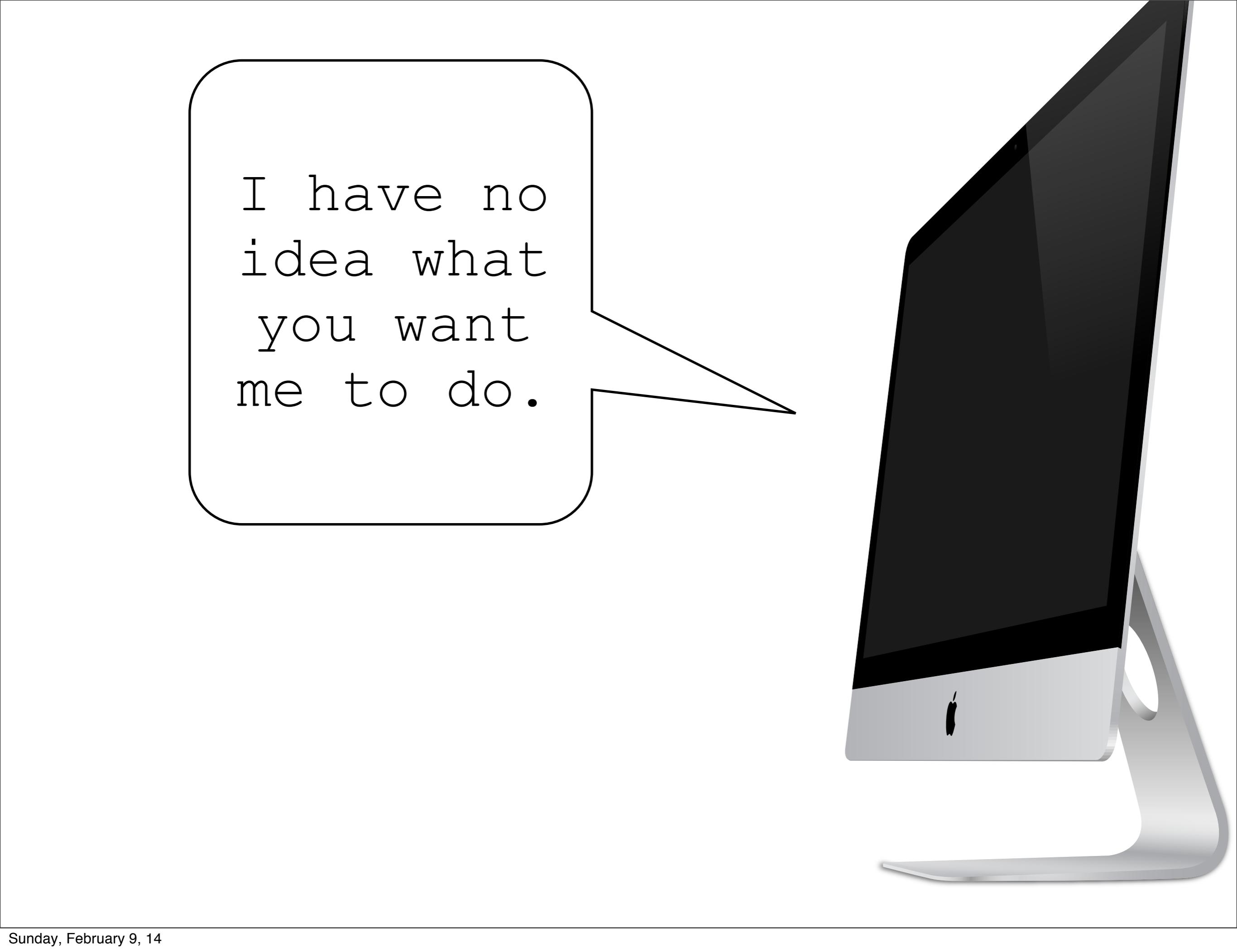
# Things that can go wrong....

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

# 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:

/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!

# **Compiler Error Demo**

# 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.

# Things that can go wrong....

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

# 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

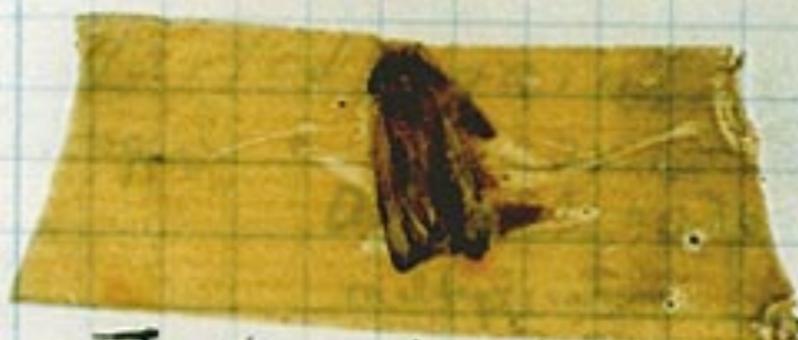
Relays 6-2 in 033 failed special speed test  
in relay " 10.000 test.

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.

# Check your assumptions.

## assumption

/ə'səm(p)SHən/ 

*noun*

plural noun: **assumptions**

1. a thing that is accepted as true or as certain to happen, without proof.  
"they made certain assumptions about the market"  
*synonyms:* [supposition](#), [presumption](#), [belief](#), [expectation](#), [conjecture](#), [speculation](#), [surmise](#), [guess](#), [premise](#), [hypothesis](#); [More](#)
  
2. the action of taking or beginning to take power or responsibility.  
"the assumption of an active role in regional settlements"  
*synonyms:* [seizure](#), [arrogation](#), [appropriation](#), [expropriation](#), [commandeering](#), [confiscation](#), [hijacking](#), [wresting](#) [More](#)





You will write bugs.



Assume it's your code.

# Things that can go wrong....

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

# Reporting Errors

- You will write code with bugs.
- Other people will write code with bugs.
- Is there anything you can do?

CPU ID: Genuine Intel 5.2.c irq1:1f SYSVER 0xf0000565

Dll Base	DateStamp	Name
30100000	3202c07e	- ntoskrnl.exe
30001000	31ed06b4	- atapi.sys
302c6000	31ed06bf	- aic78xx.sys
302d1000	31ec6c7a	- CLASS2.SYS
3c698000	31ec6c7d	- Floppy.SYS
3c90a000	31ec6df7	- Fs_Rec.SYS
3c864000	31ed868b	- KSecDD.SYS
3c6d8000	31ec6c90	- i8042prt.sys
3c874000	31ec6c94	- kbdclass.sys
3effa000	31ec6c62	- mga_mil.sys
3c708000	31ec6ccb	- Msfs.SYS
3efbc000	31eed262	- NDIS.SYS
3efa4000	31f91a51	- mga.d11
3eb8c000	31ec6e6c	- TDI.SYS
3eacf000	31f130a7	- tcpip.sys
3c550000	31601a30	- e159x.sys
3c718000	31ec6e7a	- netbios.sys
3c870000	31ec6c9b	- Parallel.SYS
3c5b0000	31ec6cb1	- Serial.SYS
3ea3b000	31f7a1ba	- mup.sys

Address dword dump Build [1381]

fec32d84	80143e00	80143e00	80144000	ffdff000	00070b02
801471c8	80144000	80144000	ffdff000	c03000b0	00000001
801471dc	80122000	f0003fe0	f030eee0	e133c4b4	e133cd40
80147304	803023f0	0000023c	00000034	00000000	00000000

Dll Base	DateStamp	Name
80010000	31eee6c52	- hal.dll
80006000	31ec6c74	- SCSIPORT.SYS
802cd000	31ed237c	- Disk.sys
8037c000	31eed0a7	- Ntfs.sys
fc6a8000	31ec6ca1	- Cdrom.SYS
fc9c9000	31ec6c99	- Null.SYS
fc9ca000	31ec6c78	- Beep.SYS
fc86c000	31ec6c97	- mouseclass.sys
fc6f0000	31f50722	- VIDEOPORT.SYS
fc890000	31ec6c6d	- vga.sys
fc4b0000	31ec6cc7	- Npfs.SYS
a0000000	31f954f7	- win32k.sys
fec31000	31eeda07	- Fastfat.SYS
feaf0000	31ed0754	- nbf.sys
feab3000	31f50a65	- netbt.sys
fc560000	31f8f864	- afd.sys
fc858000	31ec6c9b	- Parport.sys
fc954000	31ec6c9d	- ParUdm.SYS
fea4c000	31f5003b	- rdr.sys
fe9da000	32031abe	- svr.sys

Name
- KSecDD.
- ntoskrn
- ntoskrn
- ntoskrn

Restart and set the recovery options in the system control panel  
or the /CRASHDEBUG system start option.

do nothing?

# do nothing?



# Error Handling

*Code written to try to **handle errors** that happen so that the program **recovers** or **fails gracefully**.*

# Error Handling

*In Java, error handling is exception handling.*

# Exceptions

Exceptions are **objects**.

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

You can “**catch**” them.



# Exceptions



You don't always have to **catch** them.

# Throwing

You **throw** an exception when something bad happens.

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

# Catching

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

You **catch** an exception  
so that you can handle it.

# Catching

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

You don't have to catch all exceptions. You can let them **propagate**.

# Exceptions

main

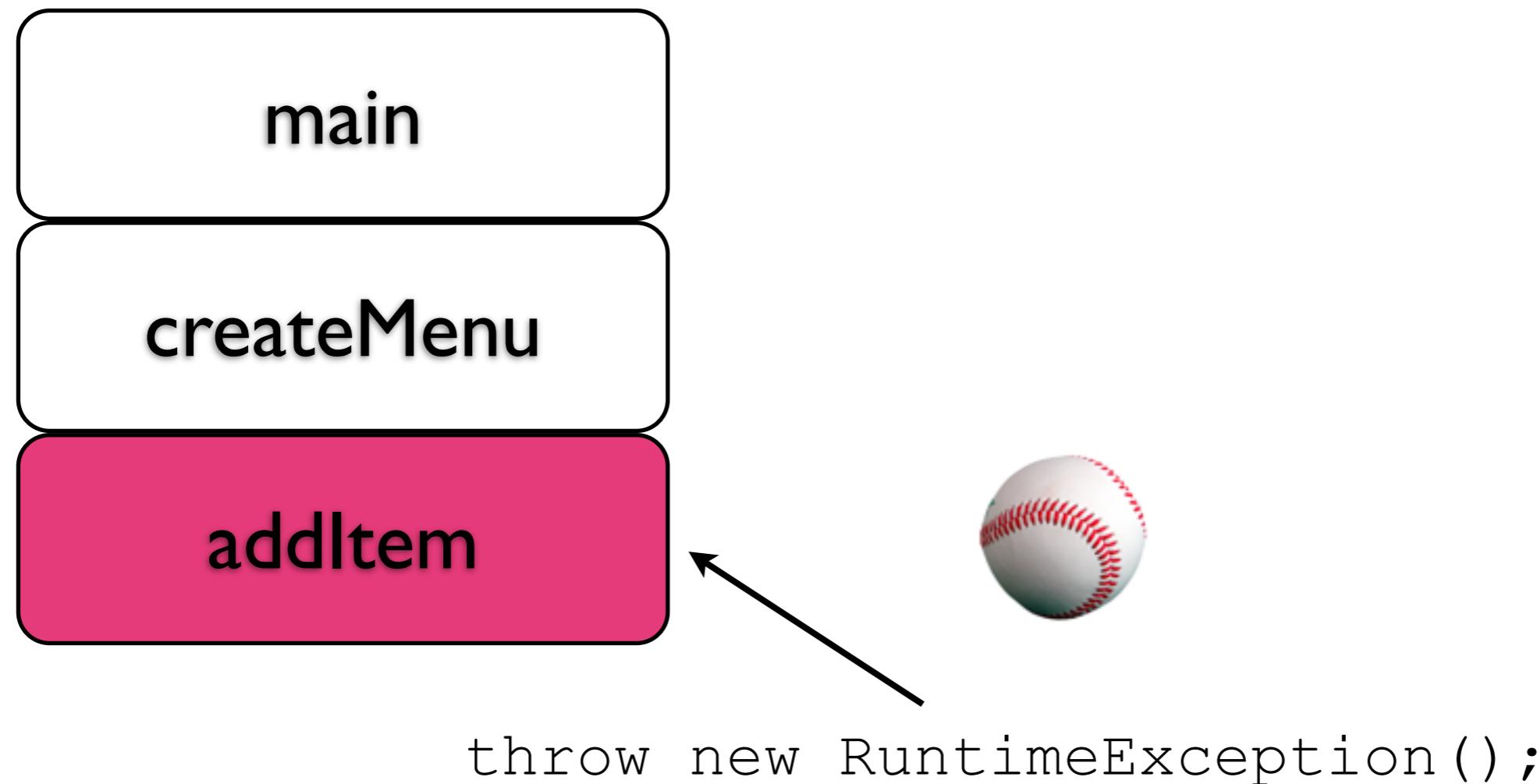
createMenu

addItem

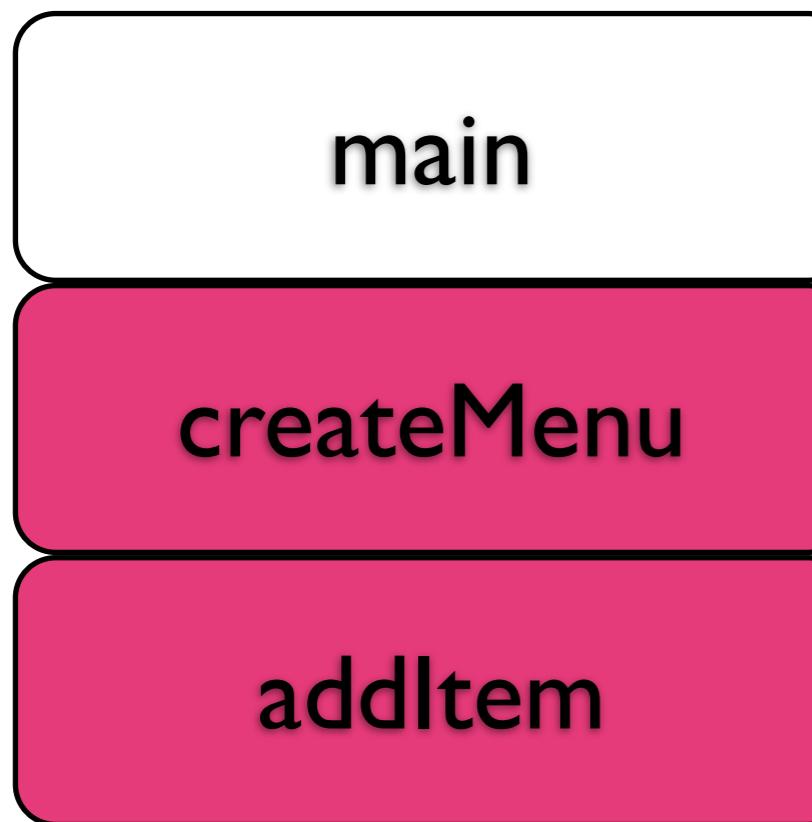
When an exception is thrown, the program immediately **jumps out** of the current function.



# Exceptions

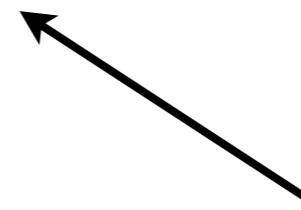
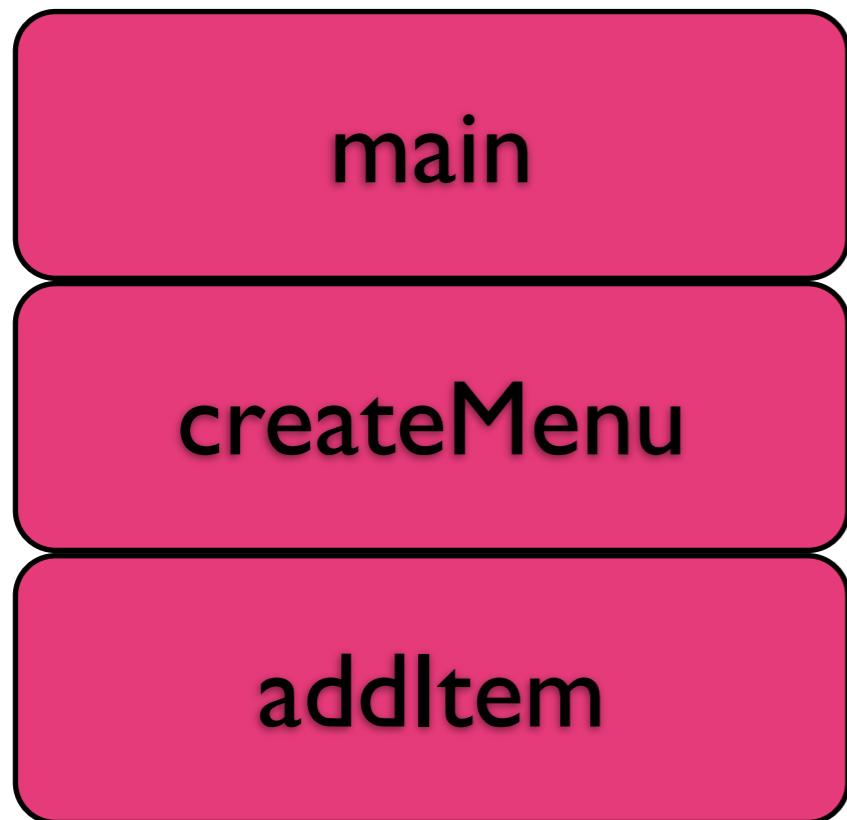


# Exceptions



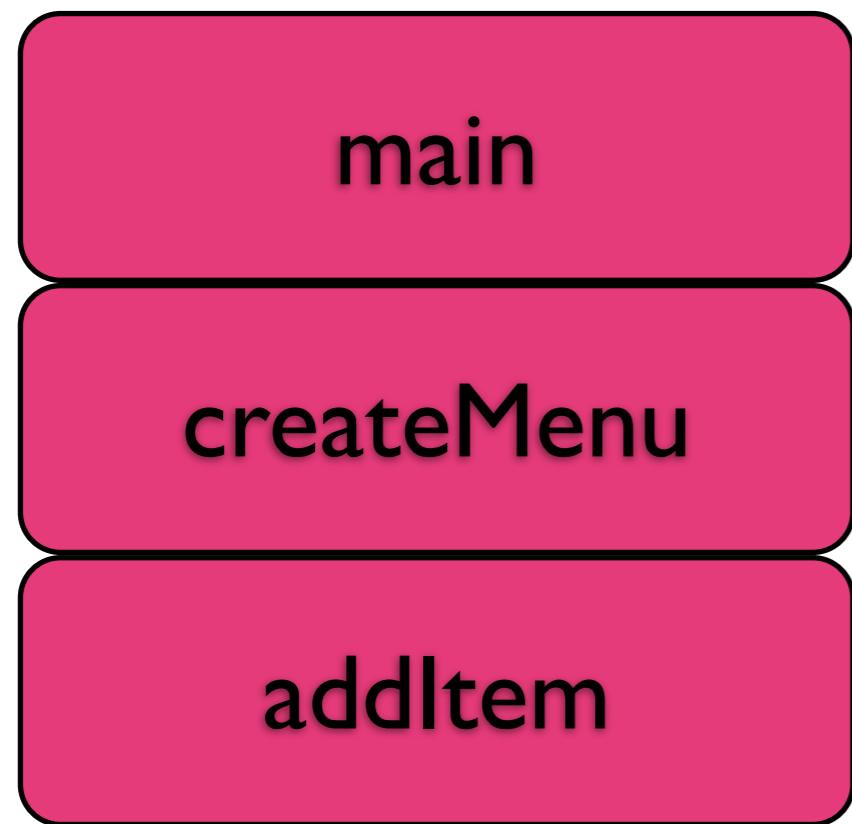
```
throw new RuntimeException();
```

# Exceptions

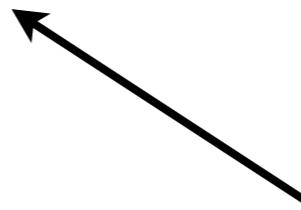


```
throw new RuntimeException();
```

# Exceptions

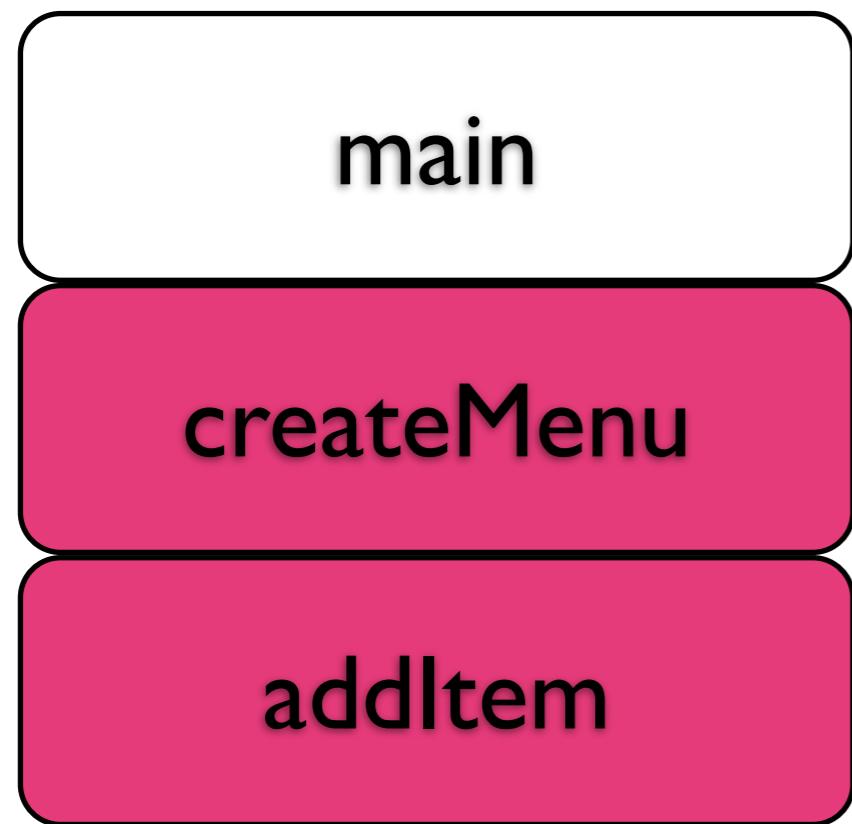


Program terminates.



`throw new RuntimeException();`

# Exceptions



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



# Why?

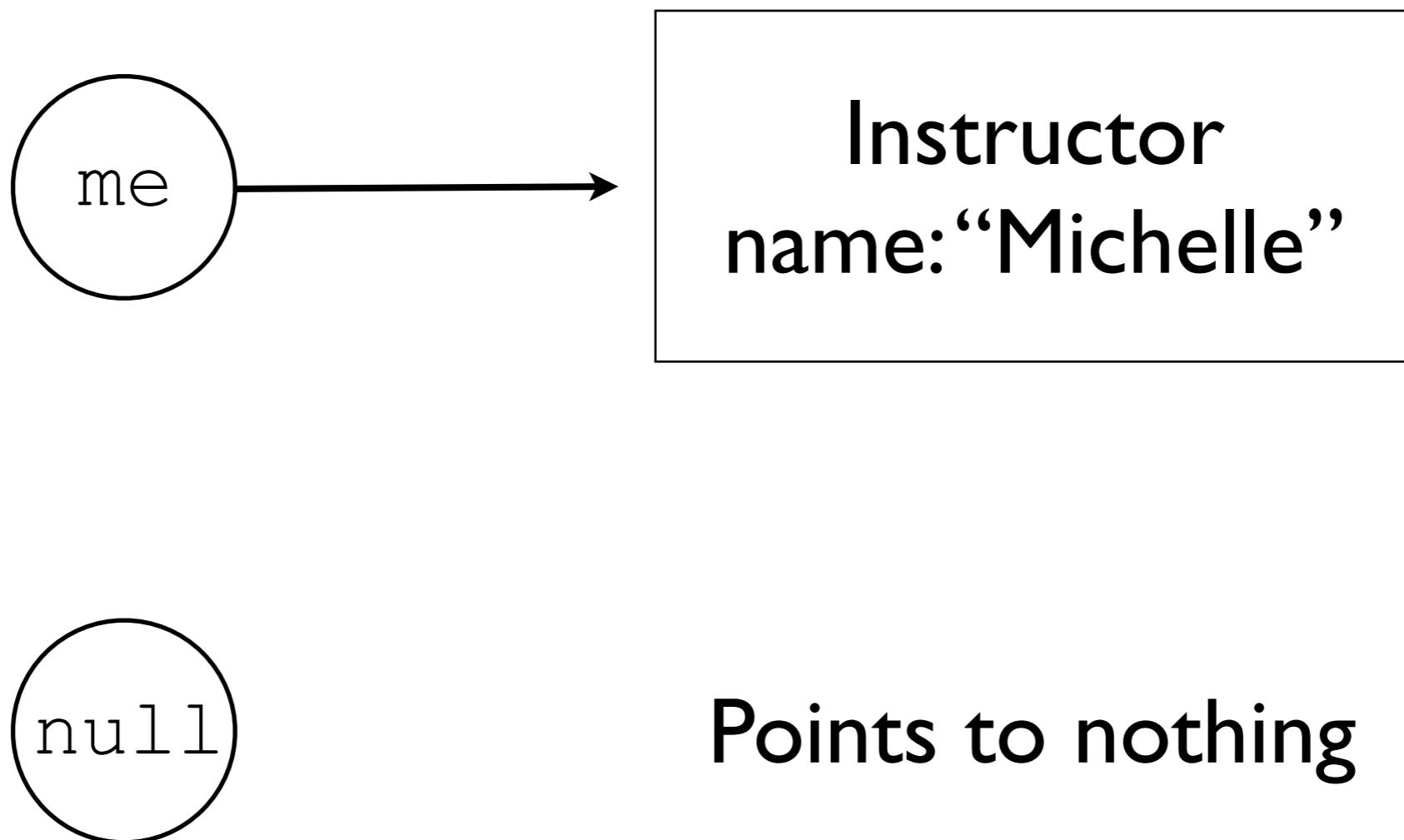
- A function gets bad parameters.
- The computer ran out of memory.
- A file couldn't be opened.
- Something didn't exist.
- Something isn't configured properly.

# 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();
```

**(getObject could return null.)**

# 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)

# What do I do?

- if I have a bug?
- if there's an exception?

**DEBUG!**

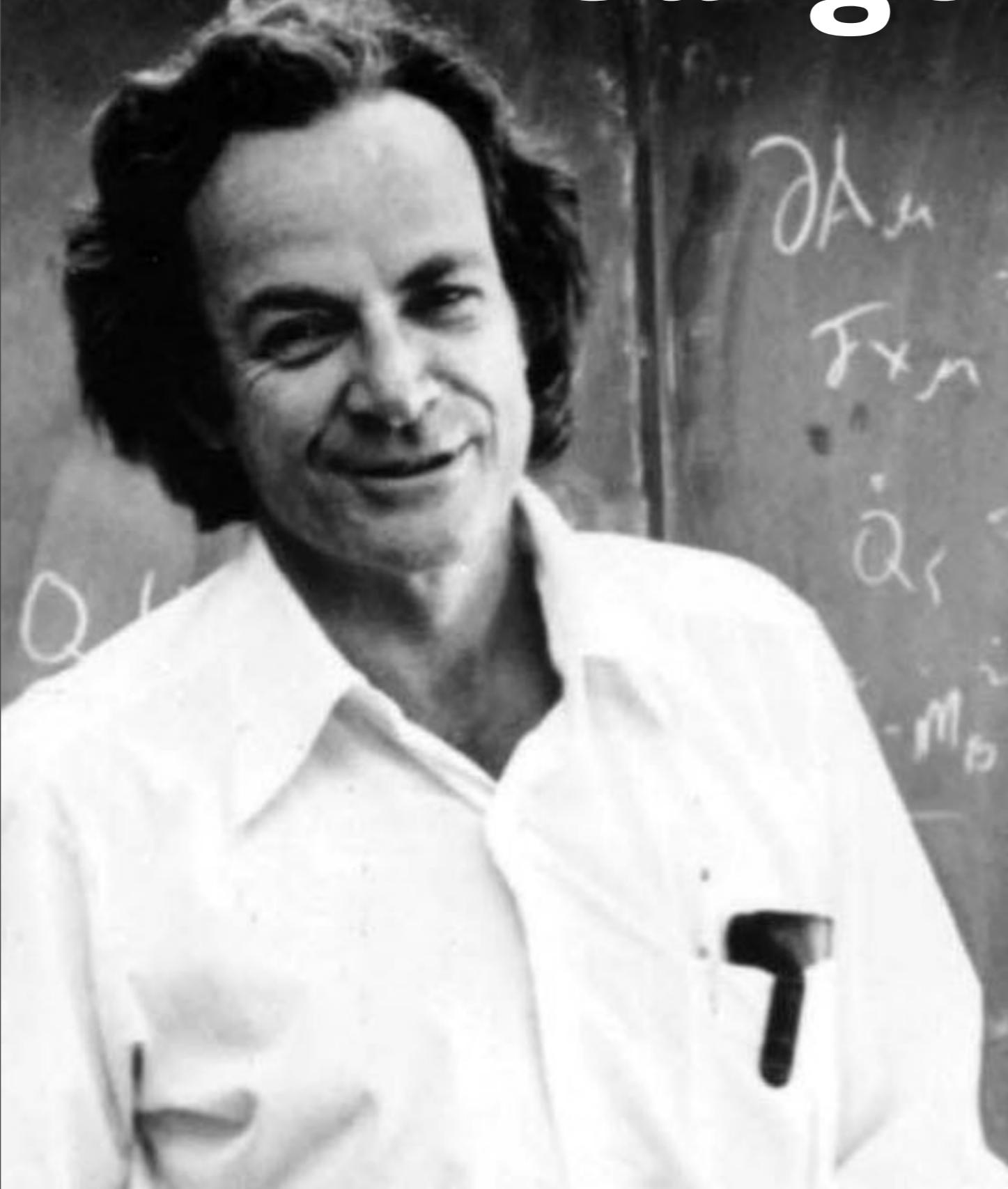
# Scientific Method



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...**

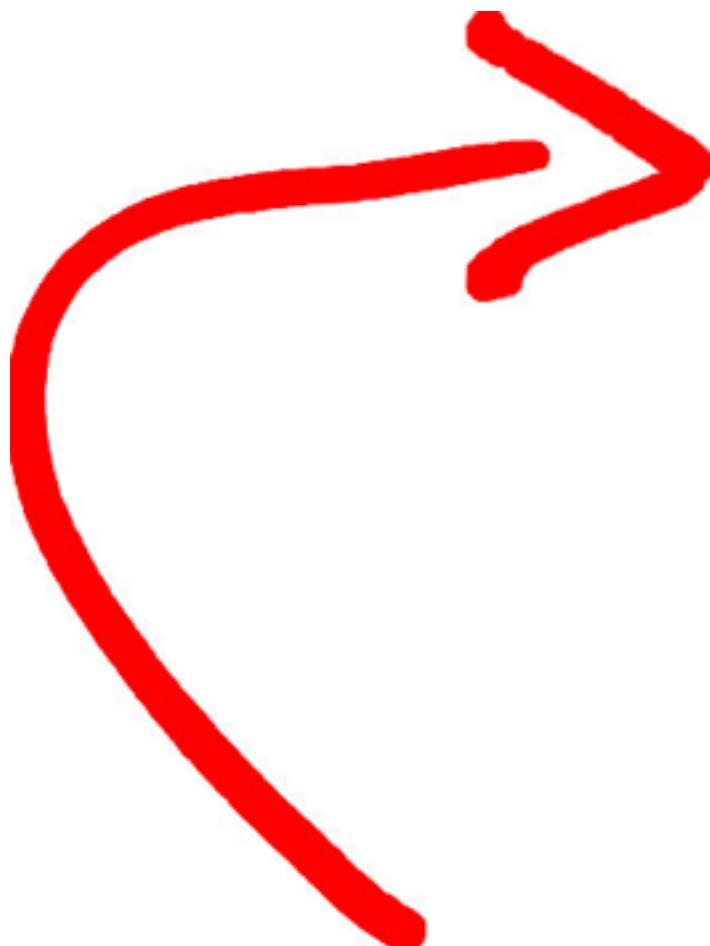
# Cargo Cult



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

Wendet Metzger Blaue  
fr. art. 040-000000  
TMA HERRN (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!**

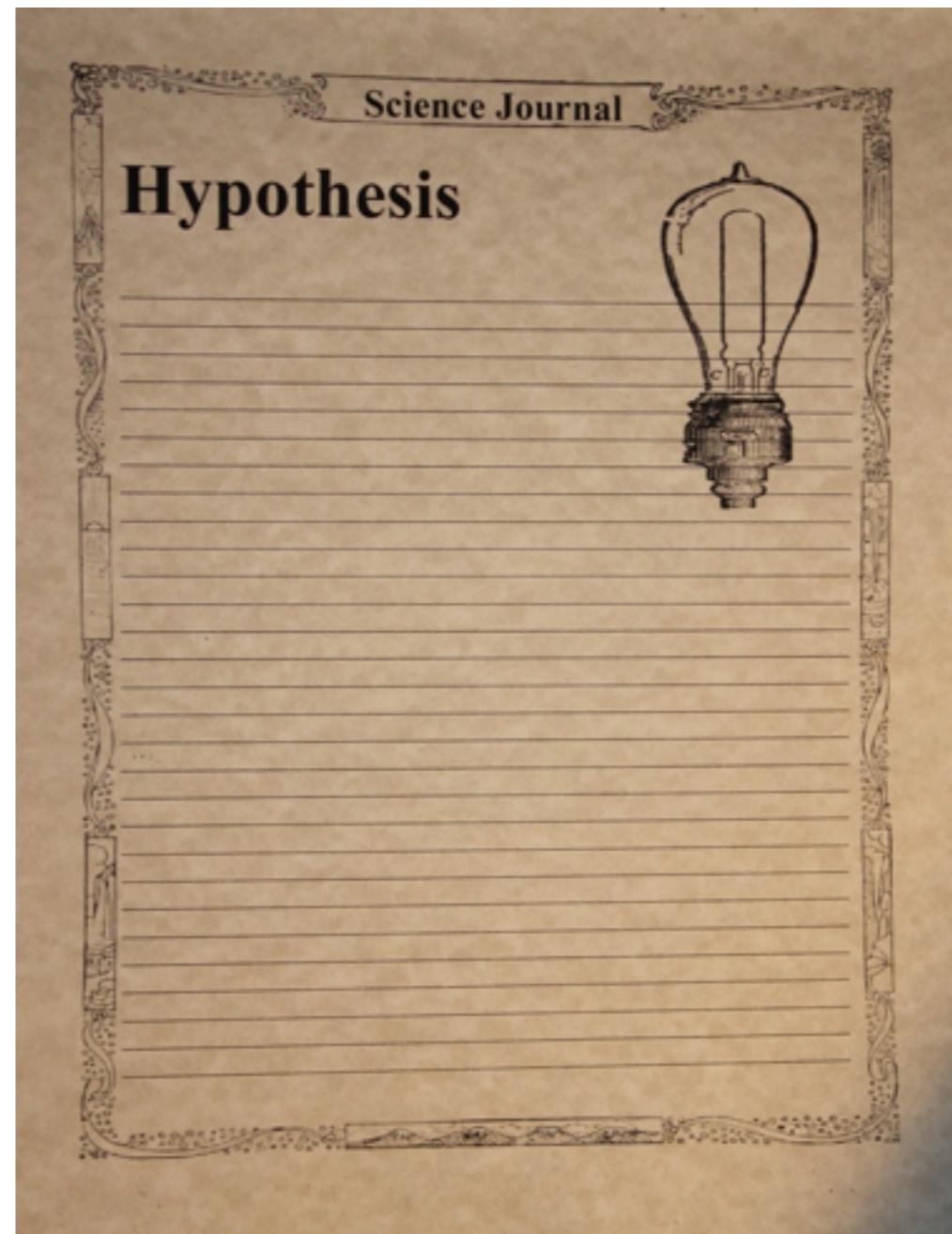
# Scientific Method



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 a "Science EXPERIMENT" worksheet. 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 section for detailing the steps of the experiment.
- Hypothesis:** A section for stating the hypothesis.
- Observations:** A section for recording observations during the experiment.
- WY DID IT HAPPEN?** A section for explaining the results.

# Design the Experiment

- Test
- Debug
- Eliminate Possibilities

# Scientific Method



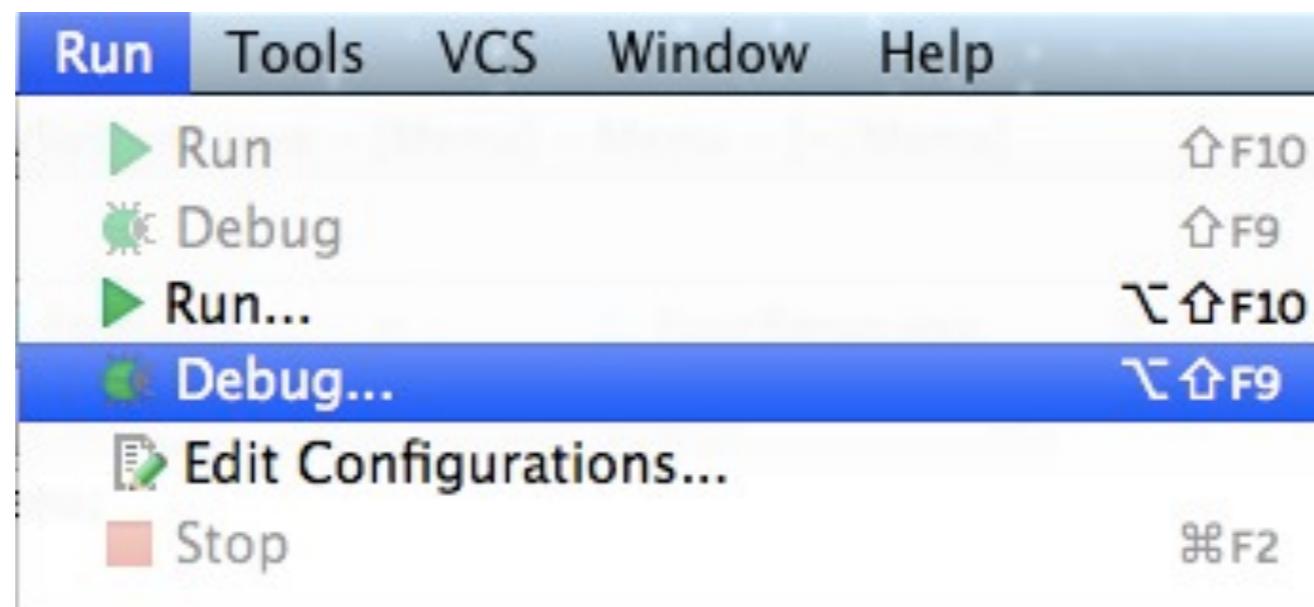
1. Characterize the problem.
2. Form a hypothesis.
3. Design an experiment.
- 4. Perform the Experiment.**
5. Evaluate the result.

# 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.*

# Debugging Demo

# Back to Exceptions



There are some  
exceptions you must  
handle.

# CheckedExceptions

- Thrown when the error could be reasonably handled by someone.
- It's a **compiler error** if you don't recognize it.
  - try/catch
  - throws

# Creating Your Own

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

# Creating Your Own

**Checked**

Exception class

**Unchecked**

RuntimeException class

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

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