# Requirements for Seminar

- Applications
  - Chrome (website: c9.io)

- GradQuant Resources
  - http://gradquant.ucr.edu/workshop-resources/

- Audience
  - No programming experience.
  - Never used Java.

# Java Fundamentals

Presented by GradQuant
Steven Jacobs

# Acknowledgement:
# Used some ideas from:

## Introduction to Python and programming

Michael Ernst

UW CSE 190p

Summer 2012

## Who should attend?

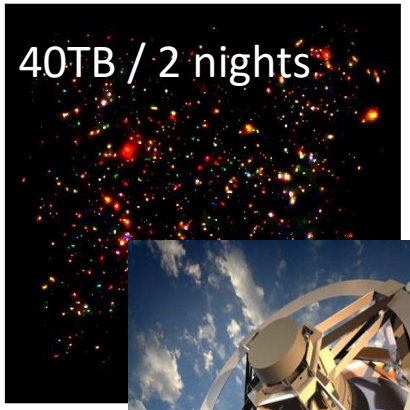
- No programing experience.
- Never used Java.
- If you came to Python Fundamentals, that's okay
  - This will follow a similar pattern but using Java code

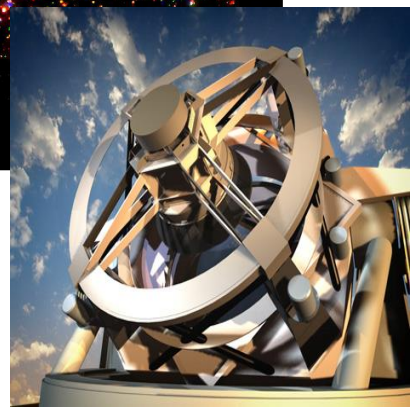
# Objectives

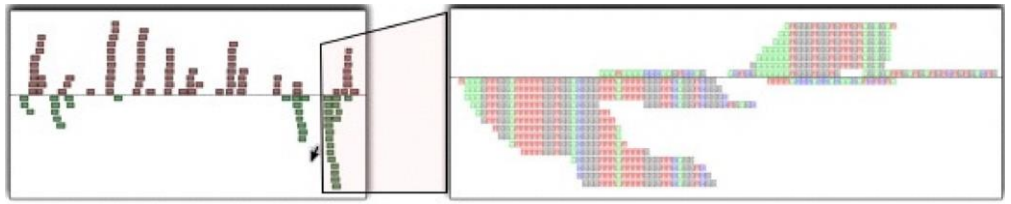
- Introduce programming concepts.
- Review Java syntax.
- Review available development tools.
- Create and compile a Java script
- Show more resources

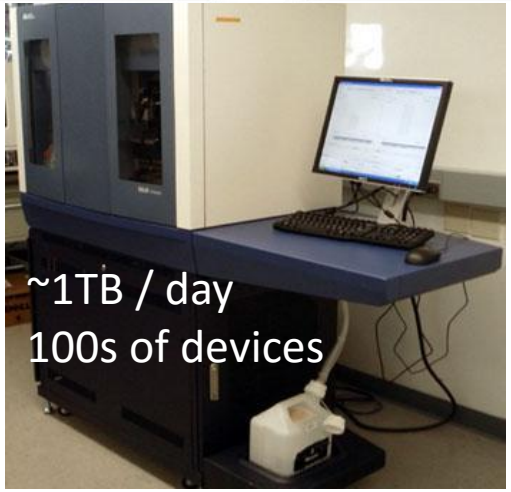
# All of science is reducing to computational data manipulation

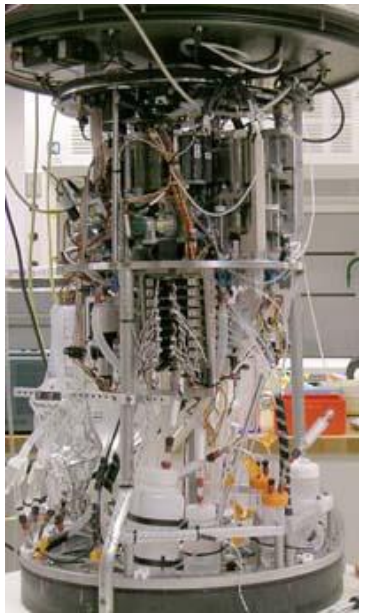
- Astronomy: High-resolution, high-frequency sky surveys (SDSS, LSST, PanSTARRS)
- Biology: lab automation, high-throughput sequencing,
- Oceanography: high-resolution models, cheap sensors, satellites

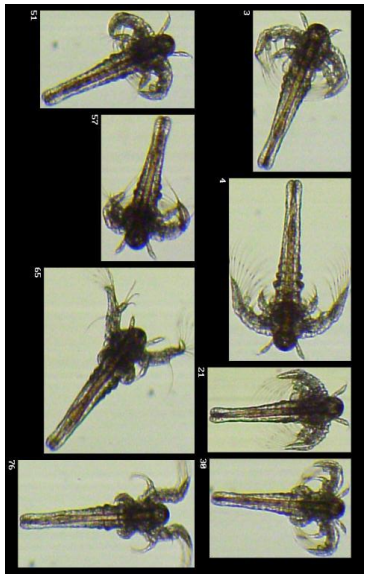








# Example: Assessing treatment efficacy

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | fu_2wk | fu_4wk | fu_8wk | fu_12wk | fu_16wk | fu_20wk | fu_24wk | total4type_fu | clinic_zip | pt_zip |
| 2 | 1 | 3 | 4 | 7 | 9 | 9 | 9 | 12 | 98405 | 98405 |
| 3 | 2 | 4 | 6 | 7 | 8 | 8 | 8 | 8 | 98405 | 98403 |
| 4 | 0 | 0 | | | | 0 | 0 | | 98405 | 98445 |
| 5 | 3 | 4 | | | | 5 | 5 | | 98405 | 98332 |
| 6 | 0 | 0 | | | | 0 | 0 | | 98405 | 98405 |
| 7 | 2 | 2 | | | | 2 | 2 | | | 98402 |
| 8 | 1 | 2 | 5 | 6 | 8 | 10 | 10 | 14 | 98405 | 98418 |
| 9 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 98499 | 98406 |
| 10 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 6 | 98405 | 98404 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 98405 | 98402 |
| 12 | 1 | 1 | 2 | 2 | 4 | 4 | 4 | 4 | 98405 | 98405 |
| 13 | 1 | | | | | | | | 98404 | 98404 |
| 14 | 2 | | | | | | | | 98499 | 98498 |
| 15 | 0 | | | | | | | | 98499 | 98445 |
| 16 | 1 | | | | | | | | 98499 | 98405 |
| 17 | 1 | | | | | | | | 98499 | 98498 |
| 18 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 98499 | 98499 |
| 19 | 1 | 1 | 4 | 5 | 7 | 7 | 7 | 7 | 98499 | 98371 |

number of follow ups within 16 weeks after treatment enrollment.

Zip code of clinic

Zip code of patient

*Question: Does the distance between the patient's home and clinic influence the number of follow ups, and therefore treatment efficacy?*

# Program to assess treatment efficacy

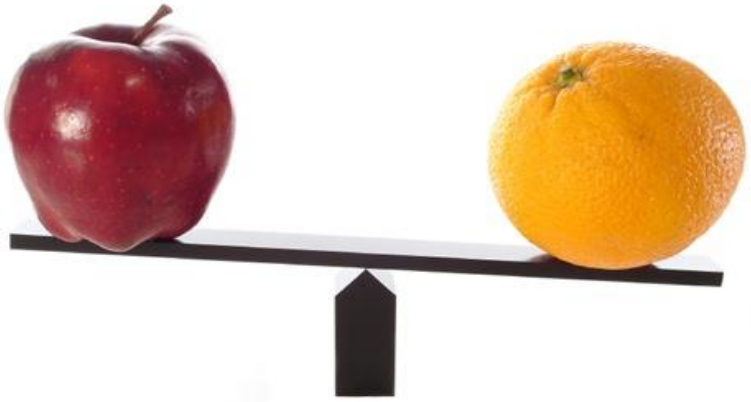
```
# This program reads an Excel spreadsheet whose penultimate
# and antepenultimate columns are zip codes.
# It adds a new last column for the distance between those zip
# codes, and outputs in CSV (comma-separated values) format.
# Call the program with two numeric values:  the first and last
# row to include.
# The output contains the column headers and those rows.

# Libraries to use
import random
import sys
import xlrd         # library for working with Excel spreadsheets
import time
from gdapi import GoogleDirections

# No key needed if few queries
gd = GoogleDirections('dummy-Google-key')

wb = xlrd.open_workbook('mhip_zip_eScience_121611a.xls')
sheet = wb.sheet_by_index(0)

# User input:  first row to process, first row not to process
first_row = max(int(sys.argv[1]), 2)
row_limit = min(int(sys.argv[2]+1), sheet.nrows)

headers = sheet.row_values(0) + ["distance"]
print comma_separated(headers)

for rownum in range(first_row,row_limit):
   row = sheet.row_values(rownum)
   (zip1, zip2) = row[-3:-1]
   if zip1 and zip2:
      # Clean the data
      zip1 = str(int(zip1))
      zip2 = str(int(zip2))
      row[-3:-1] = [zip1, zip2]
      # Compute the distance via Google Maps
      try:
         distance = gd.query(zip1,zip2).distance
      except:
         print >> sys.stderr, "Error computing distance:", zip1,
zip2
         distance = ""
    # Print the row with the distance
    print comma_separated(row + [distance])
    # Avoid too many Google queries in rapid succession
    time.sleep(random.random()+0.5)
```

23 lines of executable code!

1. A variable contains a value

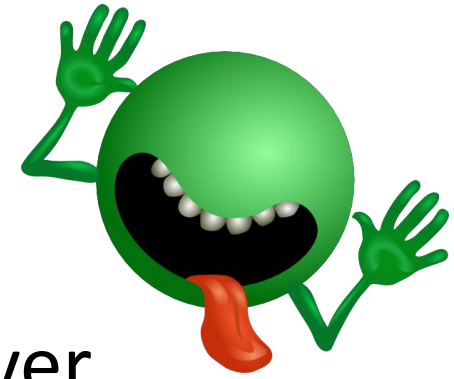
2. Java performs operations

3. Different types act differently



4. A program is a recipe

# Don’t panic!

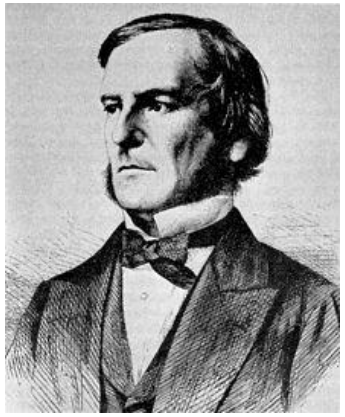


- This workshop is for people who have never programmed
  - (If you have programmed, you don’t belong here.)
- Ask questions!
  - This is the best way to learn

# 1. A variable contains a value

# Types of values (4 basic types)

- Integers (**`int`**): `-22, 0, 44`
  - No decimal points
- Real numbers (**`double`**): `2.718, 3.1415`
- Strings (**`String`**): `"I love Python”`
- Truth values (**`boolean`**): `True, False`



George Boole

# Variable Declaration

- When you declare a Java variable, you also declare it's type.
- Everything stored in that variable must be of the correct type
- Variable Declaration:

```
int x;
String myName;
double pi;
```

# Variable Assignment

- How do we store values?
  - Variables
- Assignment Operator
  - x= 5
  - NOT an equality!
  - In Java, equality is represented as ==
- "x now holds the value 5"
- In the future, x may be assigned a different value

# Declaration and Assignment

- Each variable is declared only once
- Can be assigned many times
- Can declare and assign a variable in one line
- In Java, you must assign a variable at least once before you can use it.

```
int x = 5;
int y;
String myName = "Steven"
```

# Declaration and Assignment

```
int x = 5;
int y;
String myName = "Steven";
System.out.println(x);
System.out.println(y);
6 = y;
y= x;
System.out.println(y);
x= 6;
System.out.println(x);
System.out.println(y);
myName = "Barney";
System.out.println(myName);
```

VarType.java

# Naming Rules

Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

bob  Bob  _bob  _2_bob_  bob_2  BOB

There are some reserved words:

String, int, double, class, etc.

```
int class = 5;
int String = 7;
```

# Changing existing variables ("re-binding" or "re-assigning")

```
int x = 2 - 1;
System.out.println(x);
int y = x;
System.out.println(y);
x = 5;
System.out.println(x);
System.out.println(y);
```

# Changing existing variables ("re-binding" or "re-assigning")

```
int x = 2 - 1;
System.out.println(x);
int y = x;
System.out.println(y);
x = 5;
System.out.println(x);
System.out.println(y);
```

- "=" in an assignment is *not* a promise of eternal equality
- Evaluating an expression gives a new (copy of a) number, rather than changing an existing one

reassign.java

# How an assignment is executed

1. Evaluate the right-hand side to a value
2. Store that value in the variable

```
x = 2
print x
y = x + 1
print y
x = 5
print y
z = x + 1
print x
print y
print z
```

State of the computer:

x: ~~2~~ 5

y: 3

z: 6

Printed output:

2
3
3
5
3
6

# Important: Integers vs Doubles

```
int x;
x = 2 - 1;
x = 2 * 3;
x = 2*5 + 3*4;
x = 44 / 2;
x = 14 / 4;
x = 1 / 4;
x = 3.3;
x = 14.0 / 4;
x = 14 % 4;
```

- Modulo operator (for Integers)
- 13 % 4
- 12 % 4

intExpressions.java

# Important: Integers vs Doubles

```
double x;
x = 2 - 1;
x = 2 * 3;
x = 2*5 + 3*4;
x = 44 / 2;
x = 14 / 4;
x = 1 / 4;
x = 14.0 / 4;
x = 1.0 / 4;
x = 3.3;
x = 3.7 + 10.2;
```

NOTES: Doubles can read ints, ints can't read doubles

Operation values are based on types of operands

doubleExpressions.java

# 2. Java performs operations

# Arithmetic Operations (Already seen)

```
22 * 10
22 / 10                  //return type?
22.0 / 10                //return type?
(5 +6) * (4 -3)

x = 3
y = x + 2
z = x + y
```

# More operations: Conditionals (return true/ false)

```
boolean truth;
int x;
truth = 22 > 4;
truth = 22 < 4;
truth = (x < 0);
truth = 22 == 4;
truth = (x = 100);
x = 100;
truth = (x == 200);
truth = (x == 100);
truth = (x > 5);
truth = !false;
truth = !truth;
truth = !(x >= 200);
truth = 3<4 && 7<6;
truth = 4<3 || 5<6;
int temp = 72;
boolean water_is_liquid = temp > 32 && temp < 212;
```

Operators: **!, &&, ||, <, >=, ==, !=**

**Constants: true, false**

conditions.java

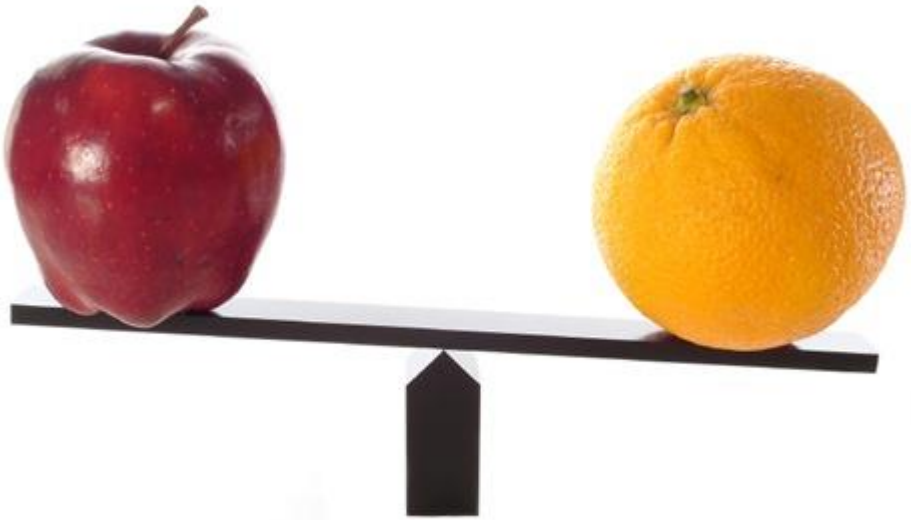
# More operations:  String

```
        String x;
        String y;
        String z= " that I have ever";
        String a= " Workshop";
        String b= " seen";
        String c= " gave the";
        String d= " best";
        x = "Steven";
        y = " Jacobs";
//Concatenation
        String sentence = x + y + c + d + a + z + b;
//Length
        int size = z.length();
//Contains
        boolean hasPH = x.contains("ph");
        boolean hasV = x.contains("v");
//Equality for Strings
        boolean isSteven = x.equals("Steven");
```

strings.java

# There are many more operations available...

# 3. Different types act differently



# Operations behave differently on different types

```
double x = 3;
double y = 2.8;
int i = 4;
int j = 5;
String a = "Doctor";
String b = "Who";
boolean one = true;
boolean two = false;
1. j = x + i;
2. y = x + i;
3. one = j + 3;
4. one = 1;
5. i = true;
6. b = a + b + i;
7. i = a + j;
```

**Which two of these will work?**

typeComparisons.java

# Type Conversion

```
double y = 2.8;
int i = 4;
String b = "42";
i = (int) y;
y = (double) i;
i = Integer.parseInt(b);
y = 3.14;
i = (int)y + Integer.parseInt(b);
```

typeConversion.java

# 4. A program is a recipe

CORNBREAD

**Colvin Run Mill Corn Bread**

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

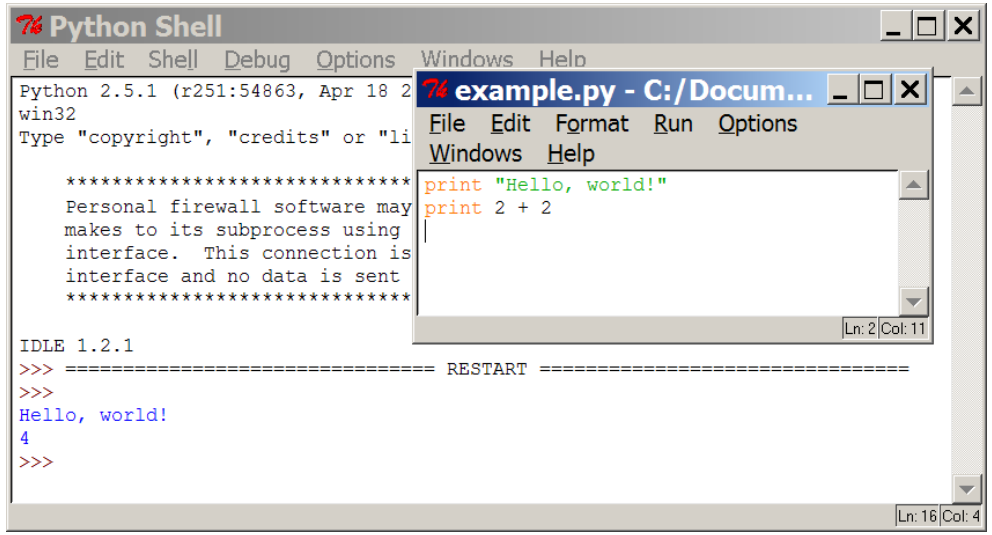
Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

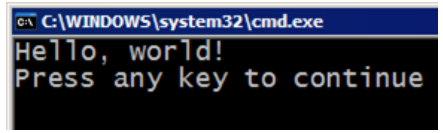
# What is a program?

- A program is a sequence of instructions
- The computer executes instructions in order

# Programming basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The place where the user interacts with the program
  - Some source code editors pop up the console as an external window, and others contain their own console window.



# White Space, curly brackets and semicolons

- White Space in ignored in Java
  - Show ugly example
- Curly Brackets open and closes classes, functions, loops, etc.
- Semicolons end statements
- Good practice
  - Indent things with tabs

uglyType.java

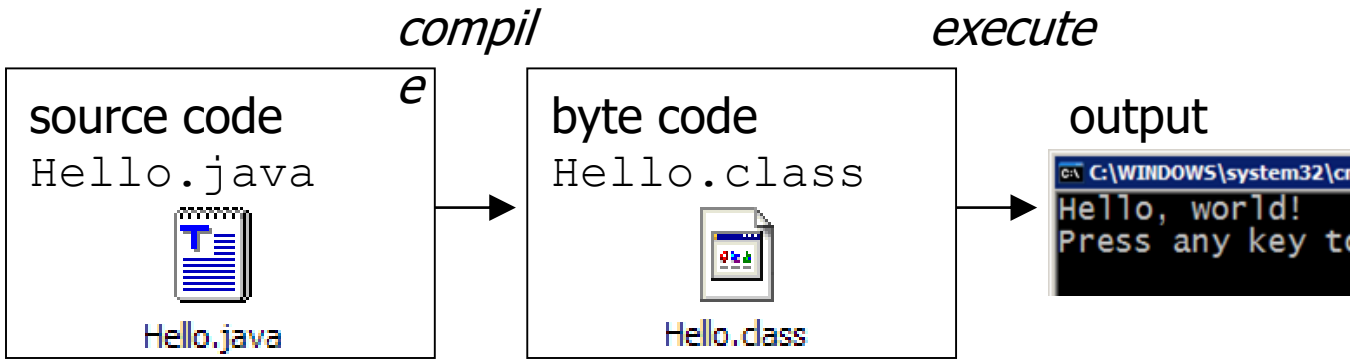
# Java Classes and "main"

- In Java, code is broken into "classes"
- A class can be seen as an application
- The "main" function within a class is the one that will execute when the class is run.
- Simple square application example
- Getting user input:
  - **System.console().readLine()**
- Converting input String to int:
  - **Integer.parseInt("34")**
- Printing to user:
  - **System.out.println("Hello John");**
  - **System.out.print("Please enter x:");**

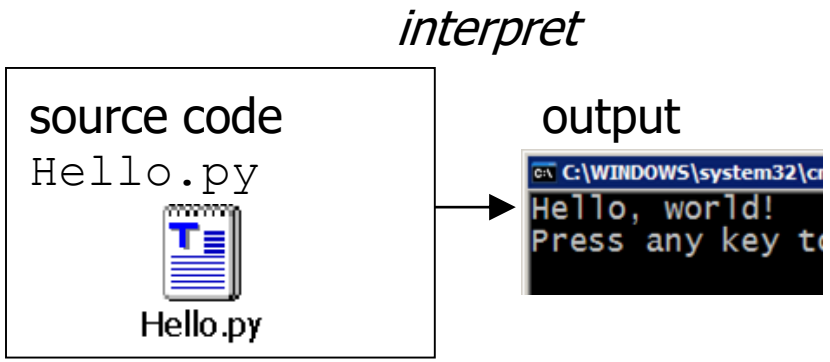
application.java

# Compiling and interpreting

- Many languages (Java) require you to *compile* (translate) your program into a form that the machine understands.




- Others (Python) turn code directly into machine instructions.




**For more on this, please attend the Python seminars!**

# Compile and Run Java

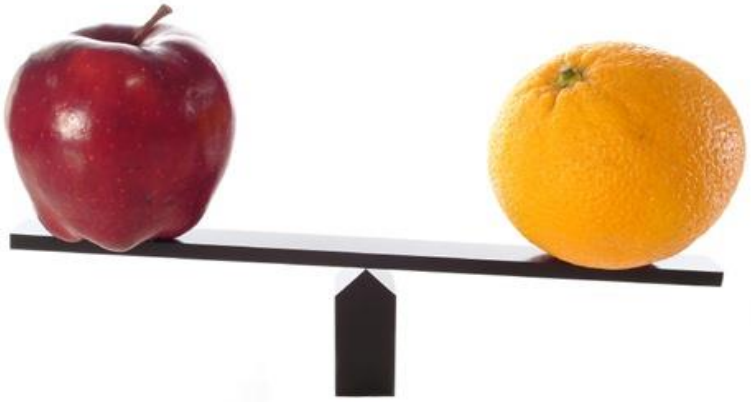
- Create a class in a .java file (e.g. application.java)
- Include a main function in your class
- Compile your code
- **javac application.java**
- Now there will be a .class file (e.g. application.class)
- Run the class (important: based on class name not file name)
- **java application**

```
class application {
    public static void main(String[] args) {
        System.out.print("Enter y:");
        int y = Integer.parseInt(System.console().readLine());
        int squared = y * y;
        System.out.println(squared);
    }
}
```

1. A variable contains a value

2. Java performs operations

3. Different types act differently



4. A program is a recipe

# Half-time!

# Comments

- Start comments with // – the rest of line is ignored.
- Can be used to document what the code is doing. This way when someone opens your code, they can see what it is for.

```
// This is a comment
```

# Software

**Online Development**

- Cloud 9 (online editor)
  - http://c9.io
  - http://bit.ly/1KIJcEU

**Local Development**

- Hello World Install/Tutorial:
  - http://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html

- Features
  - Free versions
  - Multiplatform

# Exercise 1:

```
class ex1 {
    public static void main(String[] args) {

    //Get x
    System.out.print("Enter x:");
    int x = Integer.parseInt(System.console().readLine());

    //Get y
    System.out.print("Enter y:");
    int y = Integer.parseInt(System.console().readLine());


     //add them
     int z = x + y;

     //output the result to the user
     System.out.println("x + y =");
     System.out.println(z);
  }
}
```

# Exercise 2: Fahrenheit to Celsius:

**How could we take as input from the user a Fahrenheit temperature, and then convert it to Celsius?**

**Mathematical Equation for Celsius:**

$(F - 32) \times 5/9$

**Think about:**

**Input and output**

**Integers vs Floats**

# Exercise 2:

```
class ex2 {
    public static void main(String[] args) {

        //Get farenheit
        System.out.print("Provide the temperature in Fahrenheit:");
        double f = Integer.parseInt(System.console().readLine());



        //compute celsius
        double c = (f-32) * 5 / 9;

        //output the result to the user
        System.out.println("The temperature in Celsius is:");
        System.out.println(c);
    }
}
```

# Exercise 3 (If Statement):

```
"if" provides a means of checking whether some condition is met.
Curly brackets contain what should run if the condition is met
if (5 < 6){
   System.out.println("Five is less than six");
}

if (x == "banana"){
   System.out.println("x is banana");
}

if (y <= z){
   System.out.println("y is less than or equal to z");
   System.out.println("therefore I cannot choose the wine in front of me");
}
```

# Exercise 3 (If Statement):

Have the user input a number. If this number is greater than 1000, output a message “Wow that is a big number!”

# Exercise 3 (If Statement):

```
class ex3 {
    public static void main(String[] args) {

        //Get number
        System.out.print("Provide a number:");
        int n = Integer.parseInt(System.console().readLine());

        if (n > 1000){
            System.out.println("Wow that is a big number!");
        }

        //alternatively:
        if (1000 < n){
            System.out.println("Wow that is a big number!");
        }
    }
}
```

# Exercise 4 (else if):

"else if" provides a means to check alternate conditions:

Consider this code:

```
if (n < 5){
   System.out.println("x is pretty small");
}

if (n < 10){
   System.out.println("x is average");
}

if (n < 15){
   System.out.println("x is large");
}

if (n >= 15){
   System.out.println("x is huge!");
}
```

# Exercise 4 (else if):

else if provides a means to check alternate conditions:

Consider this code:

```
if (n < 5){
   System.out.println("x is pretty small");
}

else if (n < 10){
   System.out.println("x is average");
}

else if (n < 15){
   System.out.println("x is large");
}

else {
   System.out.println("x is huge!");
}
```

# Exercise 4 (else if):

```
Let's make a text-based adventure!


Start like this:
System.out.print("You are trapped with five dragons.(A)run (B)fight (C)make friends:");
String choice = System.console().readLine();


You should output a unique message based on whether the user types A, B, or C


How do you handle when a user types something else?


String equality?
```

# Exercise 4 (else if):

```
class ex4 {
    public static void main(String[] args) {

        //Get choice
        System.out.print("You are trapped with five dragons.(A)run (B)fight (C)make friends:");
        String x = System.console().readLine();


        //print results to the user
        if (x.equals("A")){
            System.out.println("You cannot escape. You die!");
        }
        else if (x.equals("B")){
            System.out.println("You cannot win. You die!");
        }
        else if (x.equals("C")){
            System.out.println("They do not want to be friends. You die!");
        }
        else if (x.equals("cheat")){
            System.out.println("You found the way to cheat. You win!");
        }
        else{
            System.out.println("Invalid choice. You die");
        }
    }
}
```

# Moving Forward...

There are many more tools available that we can't cover here.

If you want to move forward, the next things to look at would be:
While loops
Incrementing variables
For loops
Reading/Writing files

# Java Editors

- Eclipse
  - http://pydev.org/
- Sublime Text
  - http://www.eclipse.org/

- Why use a code editor
  - Syntax Highlighting
  - Error Detection
  - Auto-completion

# Resources

- Hello World Tutorial:
  - http://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html
- Java Documentation
  - http://docs.oracle.com/javase/7/docs/api/
- GradQuant Resources
  - http://gradquant.ucr.edu/workshop-resources/
- Google
  - Search for “java …”
- Stack Overflow website
  - http://stackoverflow.com/

# GradQuant

- One-on-one Consultations
  - Make appointment on the website
  - http://gradquant.ucr.edu
- More Seminars on Programming
  - *Python Fundamentals* (Available this quarter)
  - Data Manipulation with Python (Available this quarter)
  - Advanced Python (Offered next quarter)
  - Advanced Java (Offered next quarter)
  - SQL (Available this quarter)
    - http://gradquant.ucr.edu/workshop-resources/

Remember to fill out the seminar survey. Thank you!