CMSC 150
INTRODUCTION TO COMPUTING

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO JAVA PROGRAMMING, LIANG (PEARSON 2014)

LECTURE 1

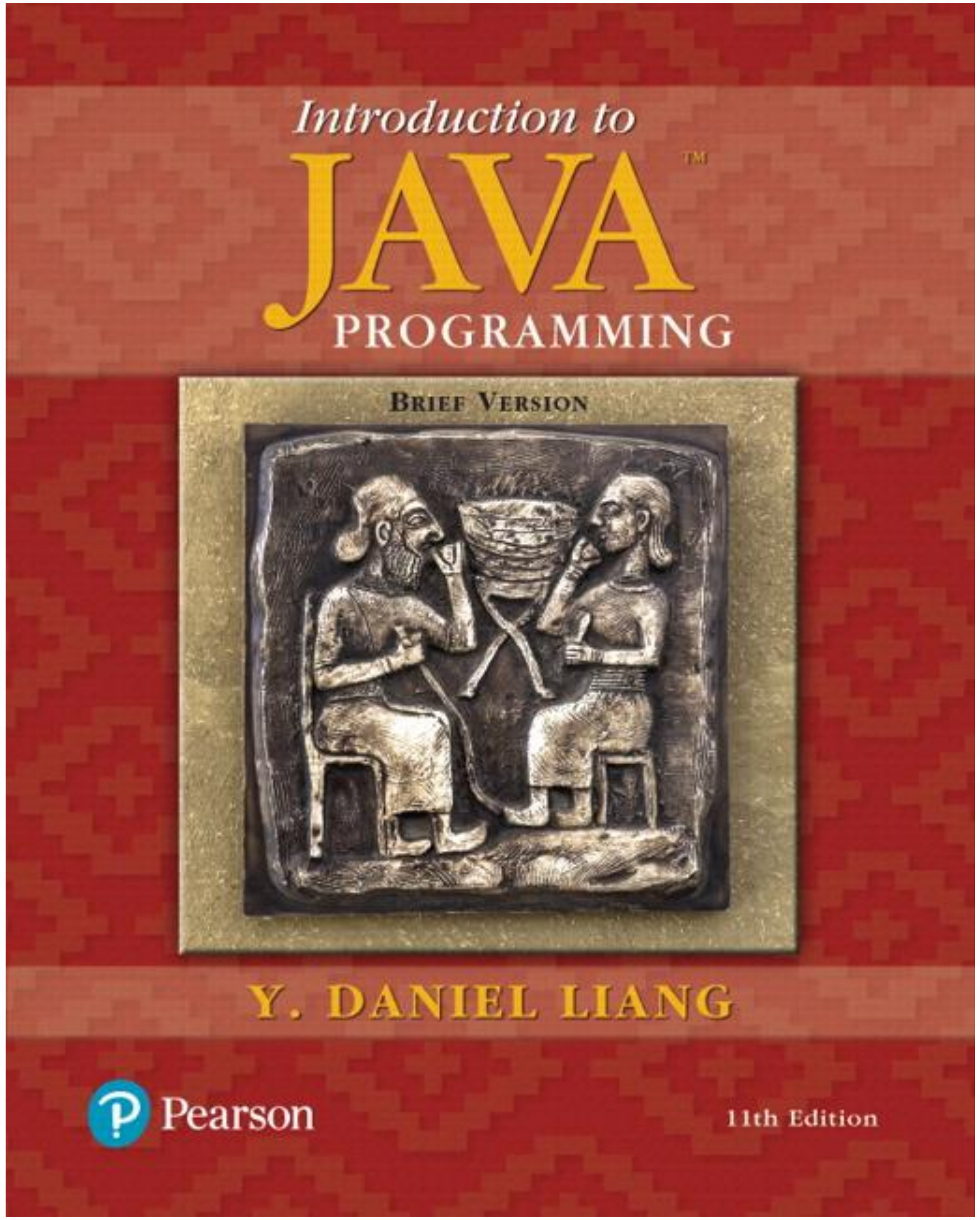
- INTRODUCTION TO COURSE
- COMPUTER SCIENCE
- HELLO WORLD

WELCOME

- Questions?

SYLLABUS

- Questions?



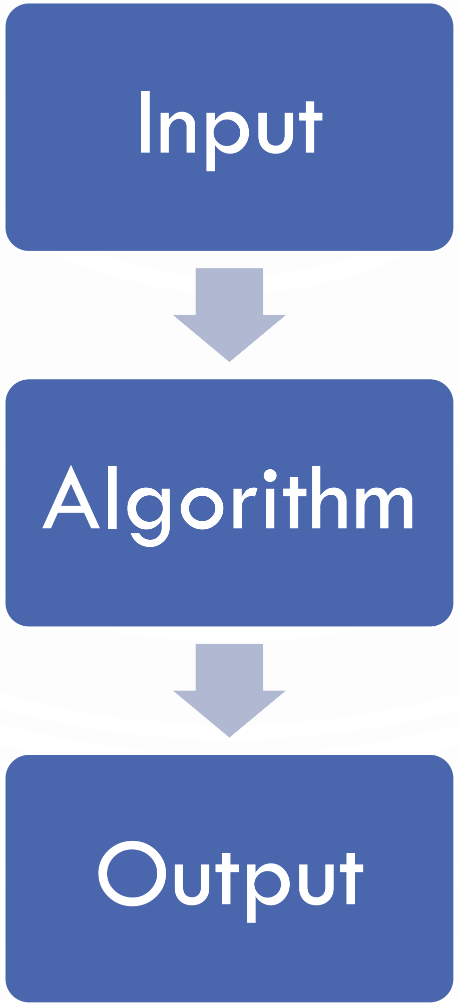
WHAT IS COMPUTER SCIENCE AND COMPUTING?

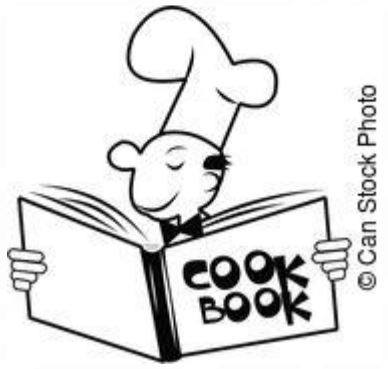
COMPUTER SCIENCE

- Your thoughts?
- Google: “The study of the principles and use of computers”
- Wikipedia: “The scientific and practical approach to computation and its applications”
- Dictionary.com: “The science that deals with the theory and methods of processing information in digital computers, the design of computer hardware and software, and the applications of computers”
- Edsgar Dijkstra: “Computer Science is no more about computers than astronomy is about telescopes”

COMPUTER SCIENCE

- Study of algorithms
- Study of computing tools
- It is not just:
  - Programming
  - Microsoft office
  - Typing
  - Electronics
  - Etc.



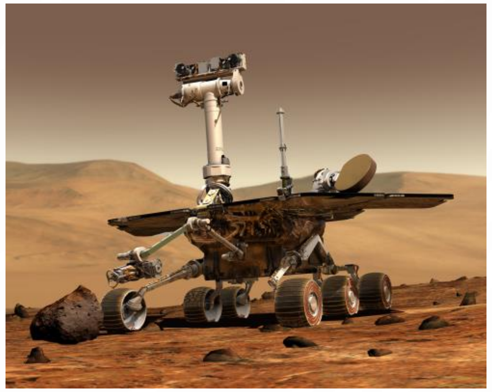


PROBLEM

- Work in pairs/triplets
- Create a methodology to perform some task, e.g.,
    - Cook something
    - Play a game
    - Buy/sell on the stock market
- Put another way…tell a computer how to do this task

PROGRAMMING

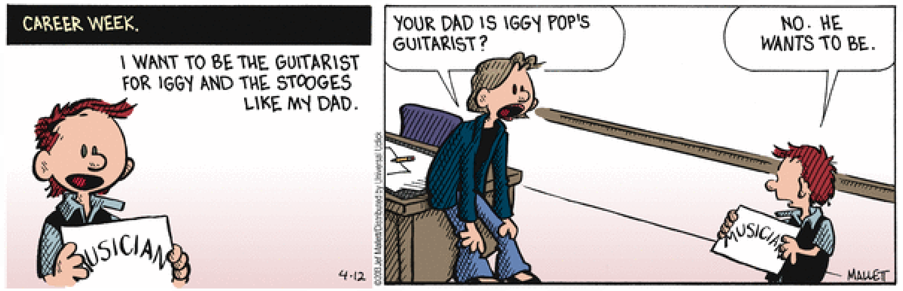
- Even though computer science is not about the computer, we still need to tell the computer what to do!
- We do this through **programming**, or the act of writing a **computer program**, known as **software** – its just instructions to the computer
- Programming allows us to push the boundaries of science, view imaginary worlds, and improve our daily lives!



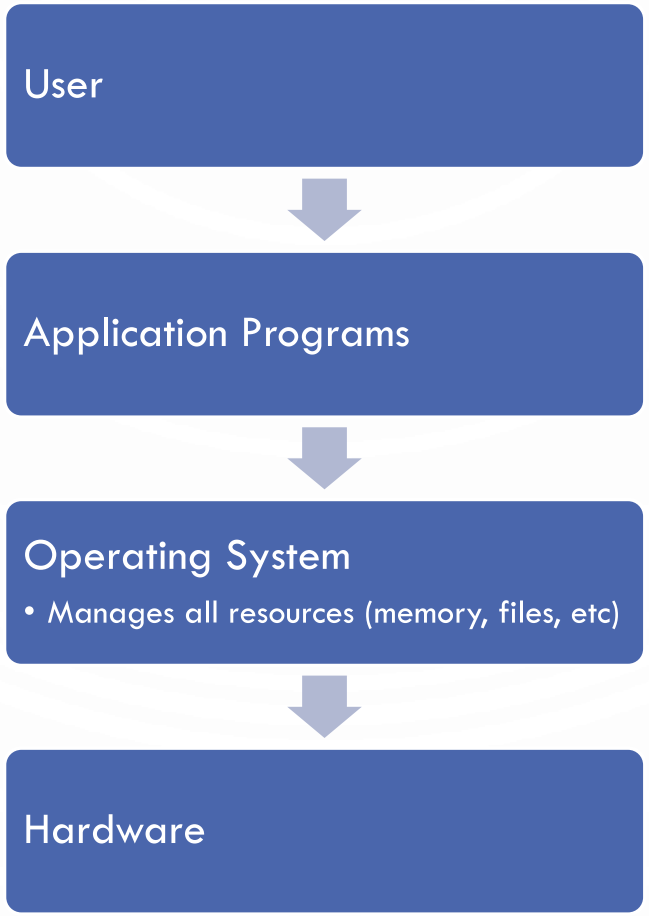
PROGRAMMING

A BRIEF NOTE ON PROGRAMMING LANGUAGES

- Machine code – 0's and 1's…or simple commands. It is the set of primitive instructions built into the computer's architecture or circuits. Extremely tedious and error prone
- Assembly code – simple commands (`ADD ra rb rc`) to make programming easier to understand. An assembler translates the commands to machine code. Extremely tedious but less error prone.
- High level languages – English-like commands that allow programming to be less tedious, less error prone, and much more expressive! Examples: Java, C++, Matlab, etc
- Why we don't use Natural language (English) – Its ambiguous…which vs which or break vs break or run vs run…ah the madness!!!!

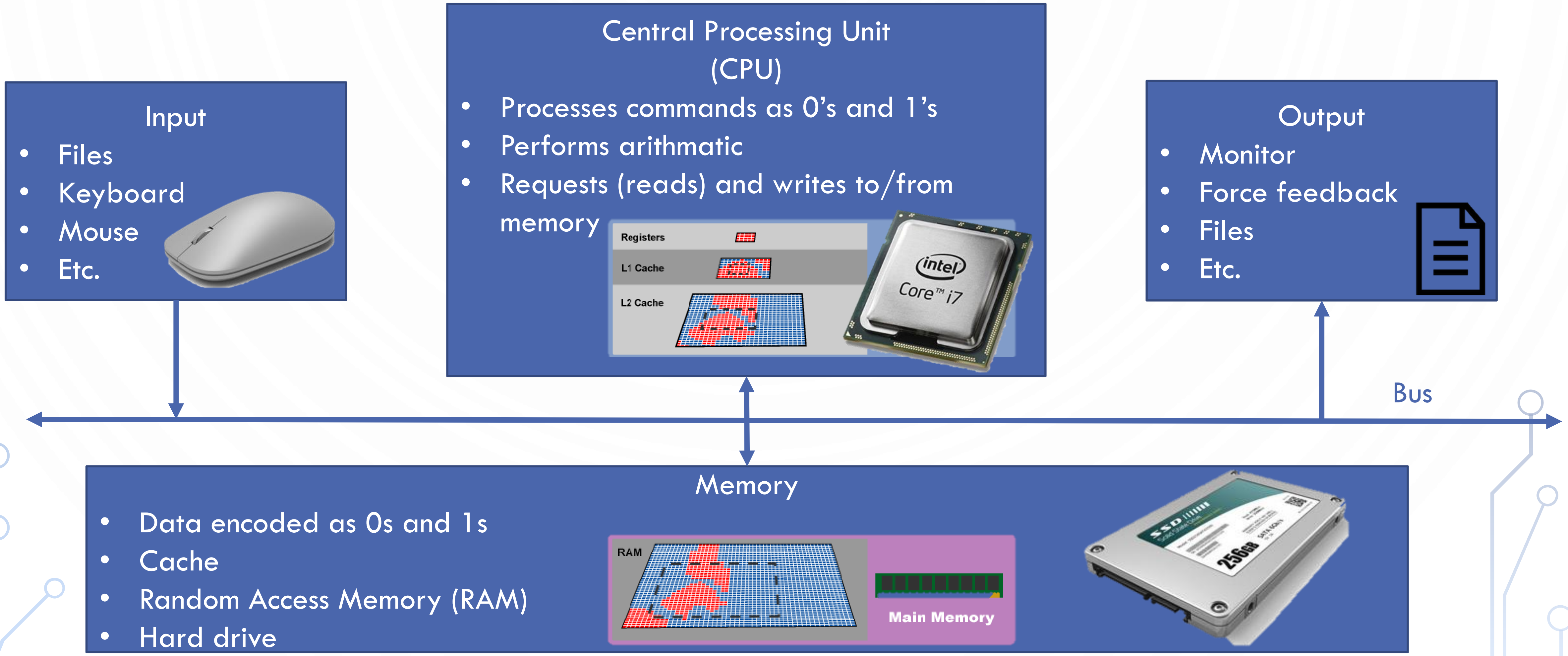

CAREER WEEK.
I WANT TO BE THE GUITARIST FOR IGGY AND THE STOOGES LIKE MY DAD.
MUSICIAN
YOUR DAD IS IGGY POP'S GUITARIST?
NO. HE WANTS TO BE.
MUSICIAN
MALLETT

COMPUTER ORGANIZATION
A SOFTWARE PERSPECTIVE



COMPUTER ORGANIZATION

A HARDWARE PERSPECTIVE



Central Processing Unit (CPU)

- Processes commands as 0’s and 1’s
- Performs arithmatic
- Requests (reads) and writes to/from memory

Input

- Files
- Keyboard
- Mouse
- Etc.

Output

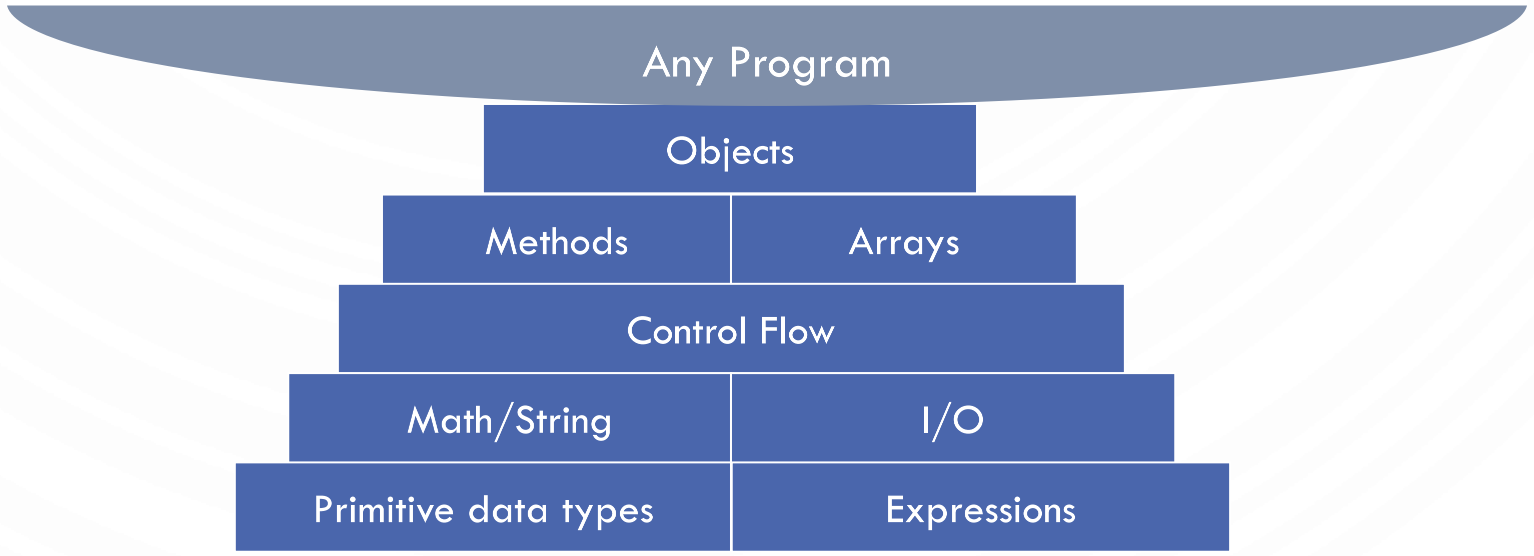
- Monitor
- Force feedback
- Files
- Etc.

Memory

- Data encoded as 0s and 1s
- Cache
- Random Access Memory (RAM)
- Hard drive

HOW DO WE PROGRAM THE COMPUTER?

- We will use Java
  - NOTE – This is an arbitrary choice. All languages build on the same basic building blocks discussed in the course. So Java is merely the vessel to our exploration of computing!
- Specifically:



WHY JAVA?

- Java
  - Widely used.
  - Widely available.
  - Embraces full set of modern abstractions.
  - Variety of automatic checks for mistakes in programs.
- Our study will
  - Minimal subset of Java.
  - Develop general programming skills that are applicable to many languages.
  - IT IS NOT ABOUT THE LANGUAGE!!!

" There are only two kinds of programming languages: those people always [gripe] about and those nobody uses."
– Bjarne Stroustrup

1.1 YOUR FIRST PROGRAM

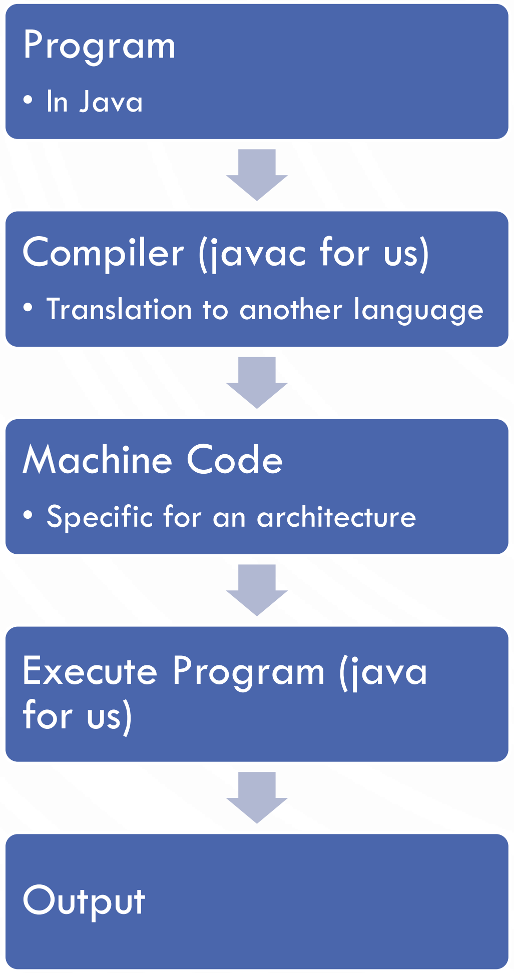
HELLO WORLD

HelloWorld.java

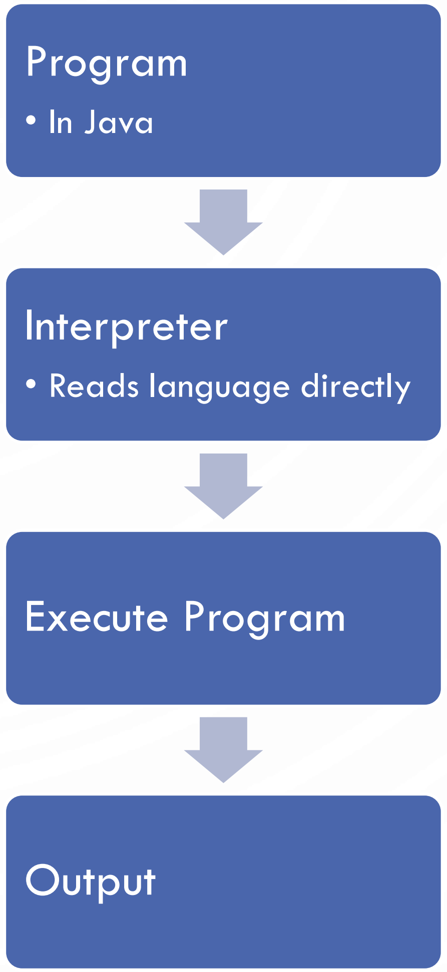
```
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println(“Hello World!");
  }
}
```

- Compile: `javac HelloWorld.java`
- Run: `java HelloWorld`

COMPILING A HIGH LEVEL PROGRAM


Program
• In Java
Compiler (javac for us)
• Translation to another language
Machine Code
• Specific for an architecture
Execute Program (java for us)
Output

Using a compiler



Using an interpreter

SUBLIME TEXT AND TERMINAL

- In this class, we will exclusively use Sublime text editor to write programs and use the terminal to compile and run our programs
- Log in
- Open a terminal
- Open sublime

TERMINAL REFERENCE GUIDE

- A **terminal** is a window to interact with your operating system through commands. Things to know:
  - You are always in a specific directory, called the **current (or working) directory**
  - Filenames are specified "relative"ly – this means you have to be in the same directory or refer to the location relative to your current directory
- Common commands (to move through folders and create them)
  - **`pwd`** – print the current directory
  - **`cd`** – change directory, e.g., **`cd`** `Desktop`
  - **`ls`** – print everything in a directory
  - **`mkdir`** – make a new directory, e.g., **`mkdir`** `HelloWorldProject`

HELLO WORLD

HelloWorld.java

```
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println(“Hello World!");
  }
}
```

- Compile: `javac HelloWorld.java`
- Run: `java HelloWorld`

ANATOMY OF A JAVA PROGRAM

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

CLASS NAME

- Every Java program must have at least one **class**. Each class has a name. By convention, class names start with an uppercase letter.
- A **class** defines an object, or entity, in your overall program. Early on we will only have one, later in the semester we will work with many.

```java
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

MAIN METHOD

- Line 2 defines the **main method**. In order to run a program, some class must contain a method named main.
- ***The program is executed from the main method.***
- Methods are subroutines meant to provide organization or package pieces for repetition.

```
1.// This program prints Hello World!
2.public class HelloWorld {
3.   public static void main(String[] args) {
4.     System.out.println("Hello World!");
5.   }
6.}
```

STATEMENT

- A **statement** represents an action or a sequence of actions.
- The statement `System.out.println(“Welcome to Java!”)` in the program in Listing 1.1 is a statement to display the greeting “Welcome to Java!”.

```java
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println(“Hello World!");
  }
}
```

STATEMENT TERMINATOR

- Every statement in Java ends with a semicolon (;).

```
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

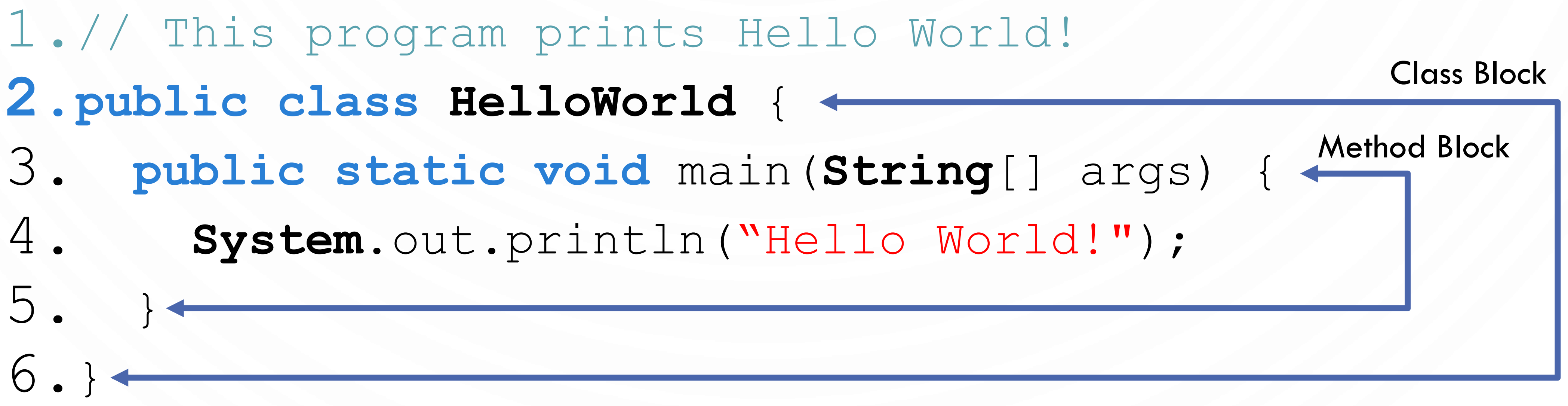
RESERVED WORDS

- **Reserved words** or **keywords** are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word class, it understands that the word after class is the name for the class.

```
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
     System.out.println("Hello World!");
  }
}
```

BLOCKS

- A pair of braces in a program forms a **block** that groups components of a program.



```
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

SPECIAL SYMBOLS

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

ASIDE, ALGORITHMIC PSEUDOCODE

- In this class, we are learning the basic tools to express and model algorithms and software. We will learn not only Java, but something called Pseudocode.
- **Pseudocode** is a detailed and stylized description for program and algorithm design. Often more natural than true language

Java code	Pseudocode
Sometimes hard to read	Stylized and easy to read
Can only use a restricted subset of math and natural language	Can use math and natural language
Generates compile errors if done improperly	Is not compiled, therefor ';', '{}', etc "don't matter"
Runs on a computer	Does not run on a computer

ASIDE, ALGORITHMIC PSEUDOCODE

JAVA CODE

HelloWorld.java

```java
// This program prints
// Hello World!
public class HelloWorld {
  public static void
      main(String[] args) {
    System.out.println(
      “Hello World!");
  }
}
```

PSEUDOCODE

HelloWorld

```
// This algorithm prints
// Hello World!
Output(“Hello World”);
```

PROGRAMMING STYLE AND DOCUMENTATION

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

APPROPRIATE COMMENTS

- Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.
- Document each variable and method
- Include your name, and a brief description at the beginning of the program.

NAMING CONVENTIONS

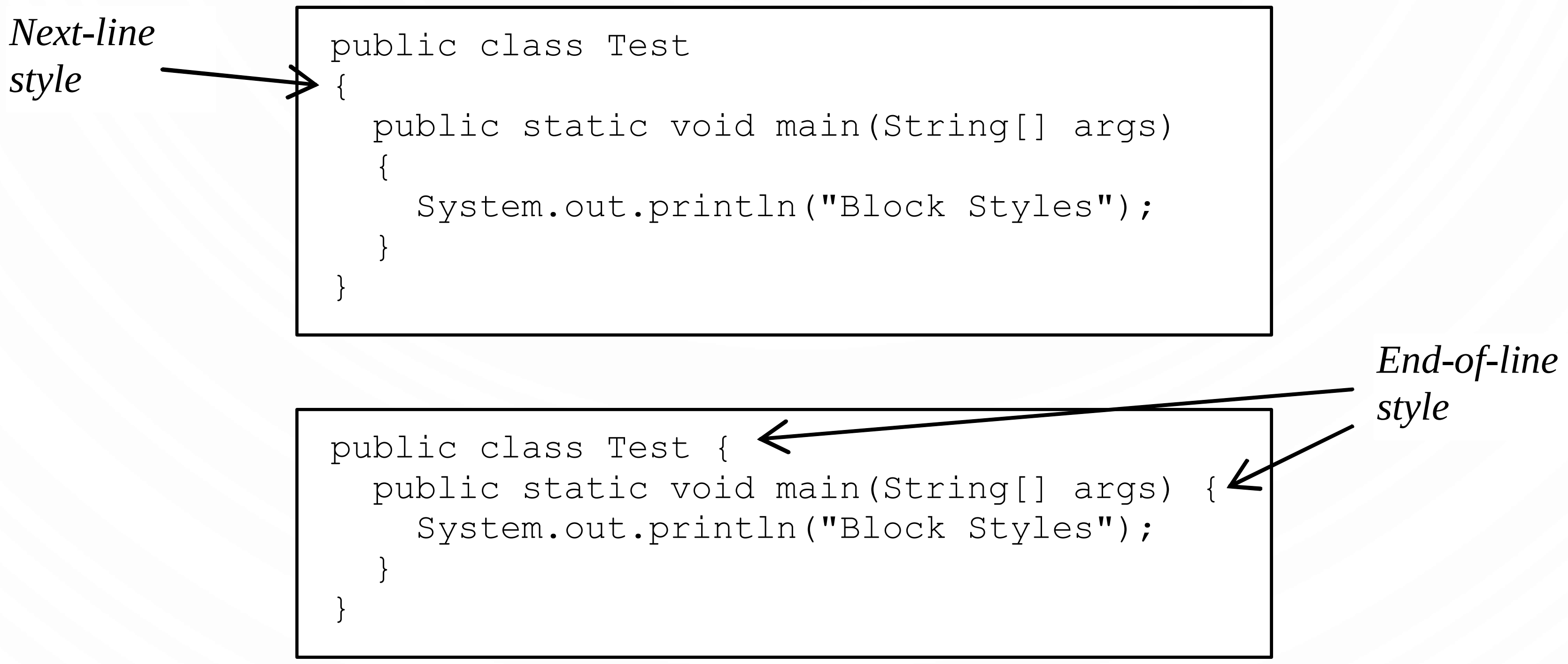
- Choose meaningful and descriptive names.
- Class names:
  - Capitalize the first letter of each word in the name, called CamelCasing. For example, the class name ComputeExpression.

PROPER INDENTATION AND SPACING

- Indentation
  - Indent two spaces.
- Spacing
  - Use blank line to separate segments of the code.

BLOCK STYLES

- Use end-of-line style for braces.



```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

PROGRAMMING ERRORS

- Syntax Errors
  - Detected by the compiler
- Runtime Errors also called Exceptions
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

SYNTAX ERRORS

- **Syntax errors** are errors from incorrectly written Java code. The compiler (**`javac`**) tells you these
- Anatomy of a compiler error:
  `filename.java:line_num: error: Confusing description of error including code where it occurs.`
- Deal with errors by experience, google, stack overflow, etc. After you have exhausted these resources…piazza/ask me. Advice, always handle the first error…not the last one.

```
// This program prints Hello World!
public Class HelloWorld {
  public static void main(String] args) {
    System.out.println("Hello World!")
  )
}
```

Can anyone spot the syntax errors?

RUNTIME ERRORS

- Runtime errors occur from impossible commands encountered while executing the program (with `java`)

```java
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println(1/0)
  }
}
```

LOGIC ERRORS

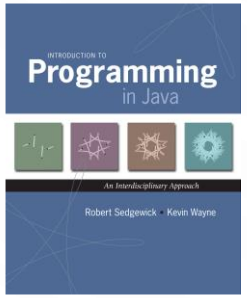
```java
// This program prints Hello World!
public class HelloWorld {
  public static void main(String[] args)
    System.out.println(
      "Celsius 35 is Fahrenheit degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

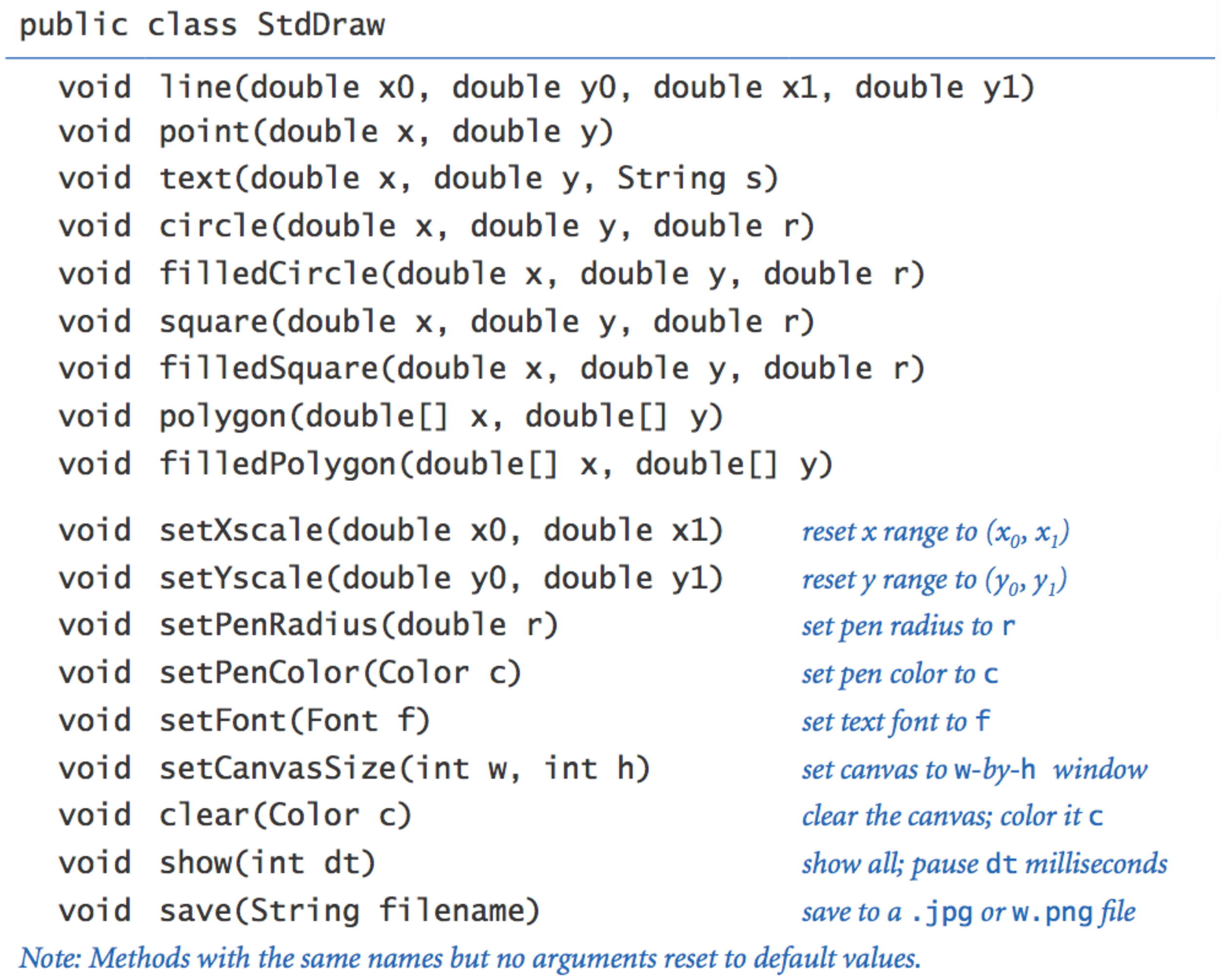
HELLO GRAPHICS!

STANDARD DRAWING

- Standard drawing (StdDraw) is library for producing graphical output

library developed
for an introduction course
(not for broad usage!)





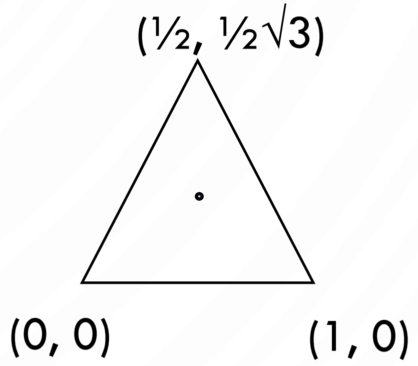
```
public class StdDraw
-----------------------------------------------------------------------------
  void  line(double x0, double y0, double x1, double y1)
  void  point(double x, double y)
  void  text(double x, double y, String s)
  void  circle(double x, double y, double r)
  void  filledCircle(double x, double y, double r)
  void  square(double x, double y, double r)
  void  filledSquare(double x, double y, double r)
  void  polygon(double[] x, double[] y)
  void  filledPolygon(double[] x, double[] y)

  void  setXscale(double x0, double x1)     reset x range to (x0, x1)
  void  setYscale(double y0, double y1)     reset y range to (y0, y1)
  void  setPenRadius(double r)              set pen radius to r
  void  setPenColor(Color c)                set pen color to c
  void  setFont(Font f)                     set text font to f
  void  setCanvasSize(int w, int h)         set canvas to w-by-h window
  void  clear(Color c)                      clear the canvas; color it c
  void  show(int dt)                        show all; pause dt milliseconds
  void  save(String filename)               save to a .jpg or w.png file
```

Note: Methods with the same names but no arguments reset to default values.

STANDARD DRAW

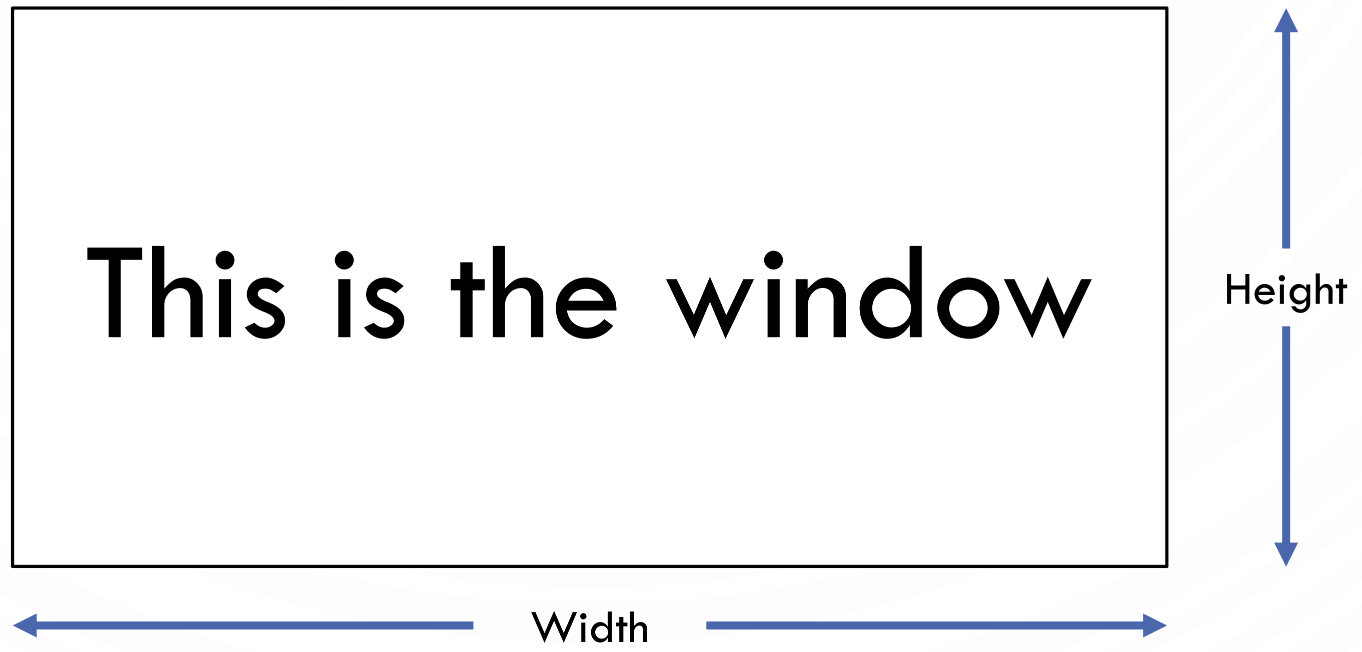
- Practice with StdDraw. To use: download StdDraw.java and put in working directory.

```java
public class Triangle {
  public static void main(String[] args) {
    StdDraw.line(0.0, 0.0, 1.0, 0.0);
    StdDraw.line(1.0, 0.0, 0.5, 0.866);
    StdDraw.line(0.5, 0.866, 0.0, 0.0);
  }
}
```



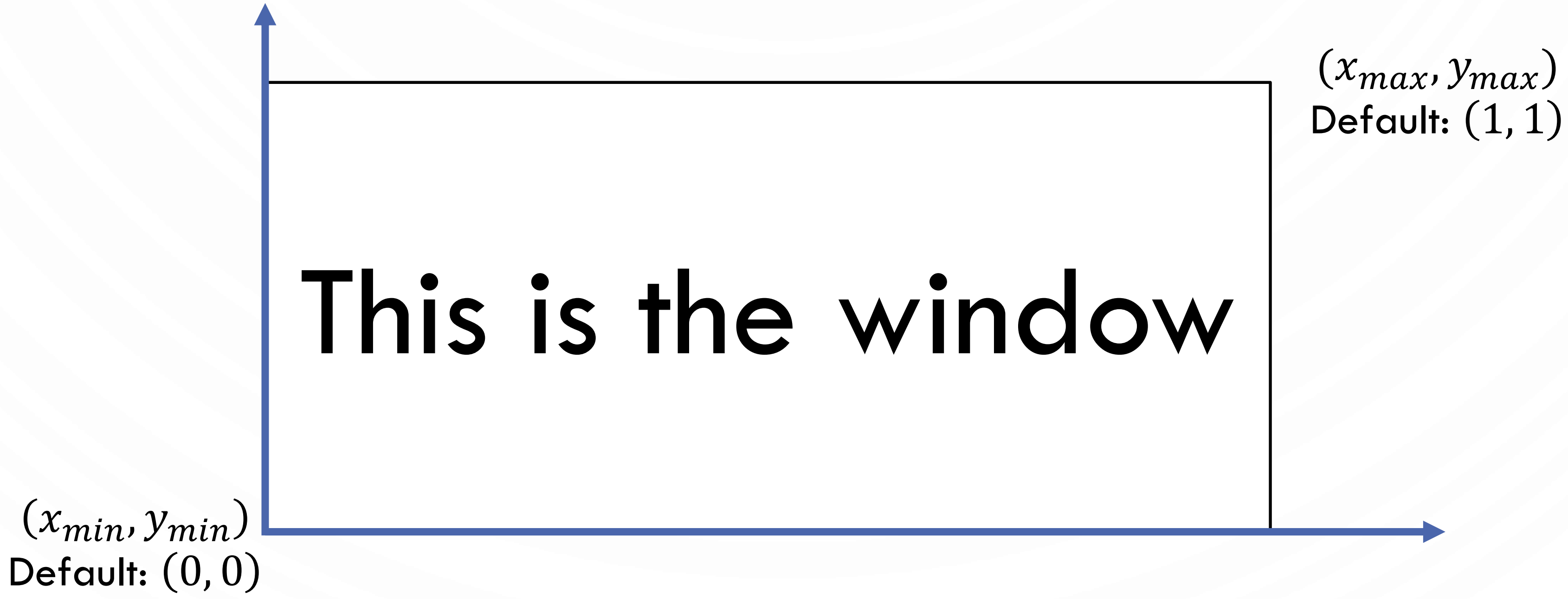
SET SIZE OF WINDOW

- Use `StdDraw.setCanvasSize(width, height)`
  - Width and height are integers representing pixels



COORDINATE SYSTEM WITH STDDRAW

- Use `StdDraw.setXscale(xmin, xmax)` and `StdDraw.setYscale(ymin, ymax)`
  - xmin, xmax, ymin, and ymax are real numbers. Note the difference between pixels!



COLORS

- Change color with `StdDraw.setPenColor(Color)`
  - Use `StdDraw.BLACK`, `StdDraw.WHITE`, `StdDraw.BLUE`, `StdDraw.RED`, etc
  - Can define own colors with Java color library (uses RGB – Red, Green, Blue)
    - `import java.awt.Color; //put at top of file`
    - `StdDraw.setPenColor(new Color(r, g, b));`

SPECIAL EFFECTS

- Images. Put .gif, .png, or .jpg file in the working directory and use `StdDraw.picture()` to draw it.

EXERCISES

1. Create a program to share three things about yourself. Please have each of the items nicely formatted with multiple System.out.println() commands.
2. Write a program using StdDraw to show a wireframe of a cube. Try to use different colors for the edges to show faces.
3. Work on Programming Assignment 1