Introduction to Java Programming
Part 2: Java, A Simple Program, and Errors

Course: CPSC 1150

Instructor: Dr. Bita Shadgar

Lecture 2

Learning Outcomes

- Recognize different features of Java
- Identify different types of Java edition, API and IDE
- Compile and execute a java program from command prompt
- Identify the anatomy of a java program
- Recognize and solve different types of error

History

- Java was developed by Sun Micro-system.
- First programming language that wasn't tied to any particular Operating System or Hardware.
- In 1990 a research group of Sun Micro-system named as Green Team was formed to create a reliable language to write control routines to a variety of electronics products.
- The new programming language named as Oak.
- Oak was not accepted by HW companies that produced controllers.
- The Green Team was a failure.

History

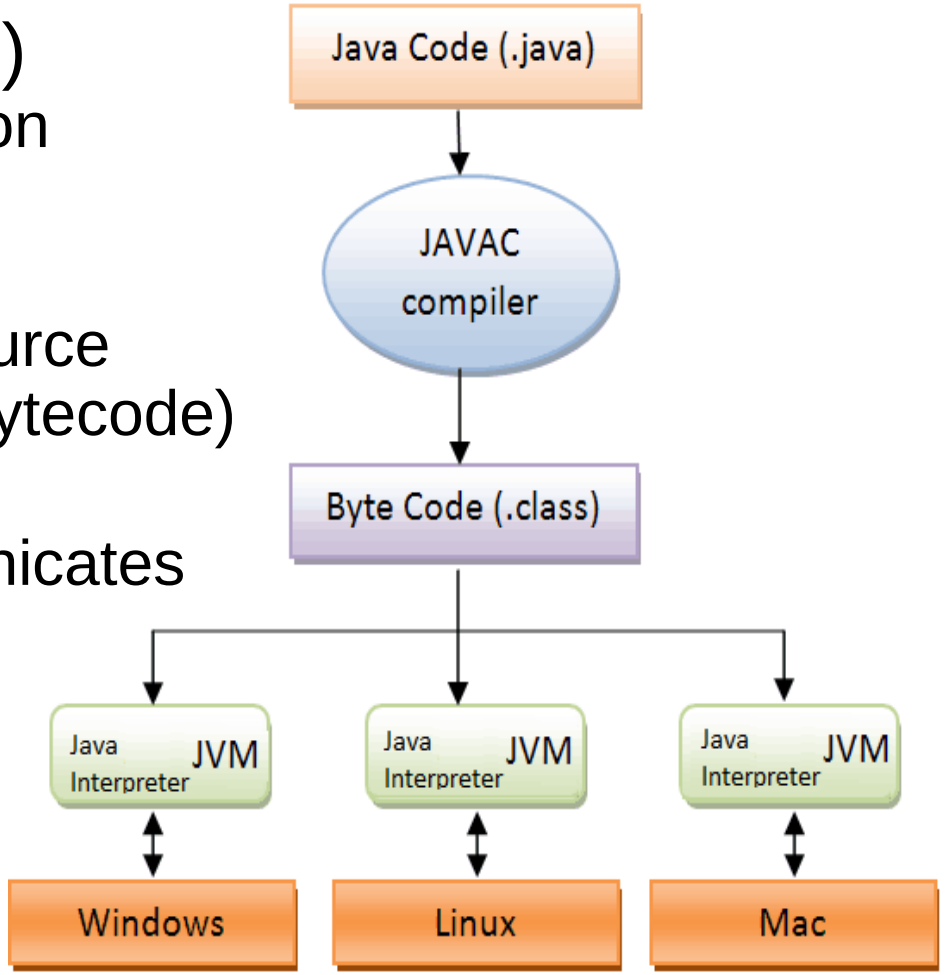
- Development of Oak was at the same time of development of World Wide Web.
- Sun came with an idea to modify Oak to a program to run on different computers.
- Oak was renamed Java in January 1995

Why Java?

- Java is a general-purpose programming language
- Java is the Internet programming language
  - Java can be used to develop Web applications.
    - Java Applets
  - Java can also be used to develop applications for hand-held devices such as Palm and cell phones

Main Features of the Java

- Java Virtual Machine (JVM)
  - Instructions are not executed on the computer directly
- Bytecode
  - Java compiler converts the source code into a binary program (bytecode)
- Java interpreter
  - Checks bytecode and communicates with the OS
  - Executes bytecode instructions line by line within the JVM



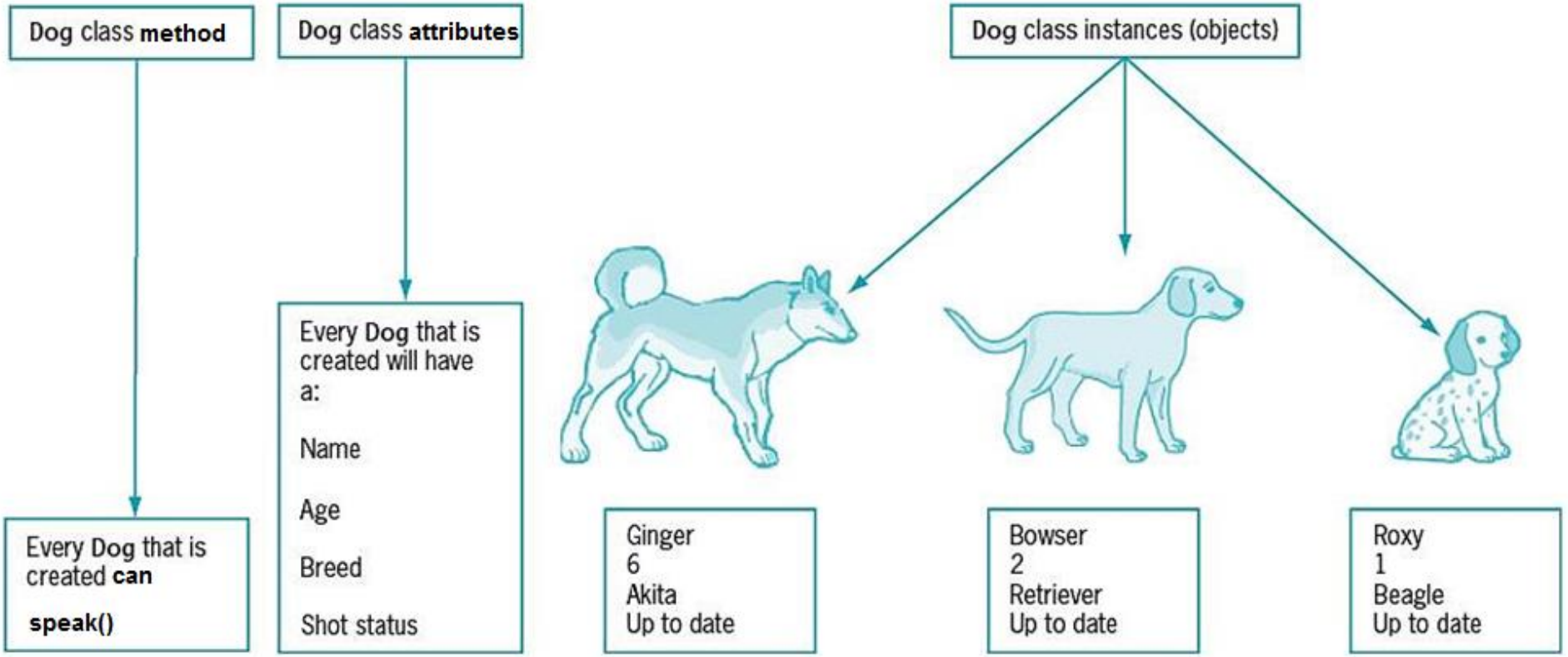
Characteristics of Java

- Java Is Simple
- Java Is Distributed
- Java Is Multithreaded
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is HW and OS independent
- Java's Performance
- Java Is Dynamic
- Java Is Object-Oriented

Classes and Objects

- Class
  - Describes objects with common properties (attributes)
  - A definition
- Objects
  - Specific, concrete instances of a class
- Attributes
  - Characteristics that define an object
  - Differentiate objects of the same class
  - The value of attributes is an object's state
- Method
  - A self-contained block of program that carries out an action
  - Similar to a function

Example – Classes and Objects



Learning about Programming

- Program - A set of instructions that you write to tell a computer what to do
- High-level languages – A language that allow programmer to use semi-natural language to write the instructions like ‘read’ , ‘for’.
- Syntax: The rules of the language.
- Semantic: Meaning of the program.

Java Program Types

- Java applets – programs embedded in a Web page
- Java applications – stand-alone programs
  - Console applications- support character output to a computer screen in a Command window
  - Event-driven applications- create a graphical user interface (GUI)

JDK Editions

- Java Standard Edition (J2SE)
  - J2SE can be used to develop client-side standalone applications or applets.
- Java Enterprise Edition (J2EE)
  - J2EE can be used to develop server-side applications such as Java servlets and Java ServerPages.
- Java Micro Edition (J2ME).
  - J2ME can be used to develop applications for mobile devices such as cell phones.

This book uses J2SE to introduce Java programming.

Java Language Specification

- Application Program Interface (API) : contains predefined classes and interfaces for developing java programs.
  - http://docs.oracle.com/javase/8/docs/api/
  - You can also download it from http://download.java.net/jdk8/docs/api/
- Integrated Development Environment (IDE) - is a java development tool. Editing, compiling, building, debugging, and online help are integrated in one GUI
  - example: NetBeans, Eclipse

A Simple Java Program

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- This is a complete Java program
- Look at it and see what kinds of words and symbols are used

A simple Java program: Classes

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- Every program must have at least one class
- All other code must be inside a class
- The first line defines a class called HelloWorld
- By convention, a class name begins with an uppercase letter

A simple Java program: The main method

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- A program always begins executing at the main method
- The main method must always be defined in the same way – we will learn more about it later
- Inside the main method is a collection of **statements**

A simple Java program: Comments

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- A one-line comment begins with a double slash, //
- A multi-line comment is enclosed between /* and */
- Comments are ignored by the computer
- Comments are there to make the program more understandable to programmers

A simple Java program: Statements

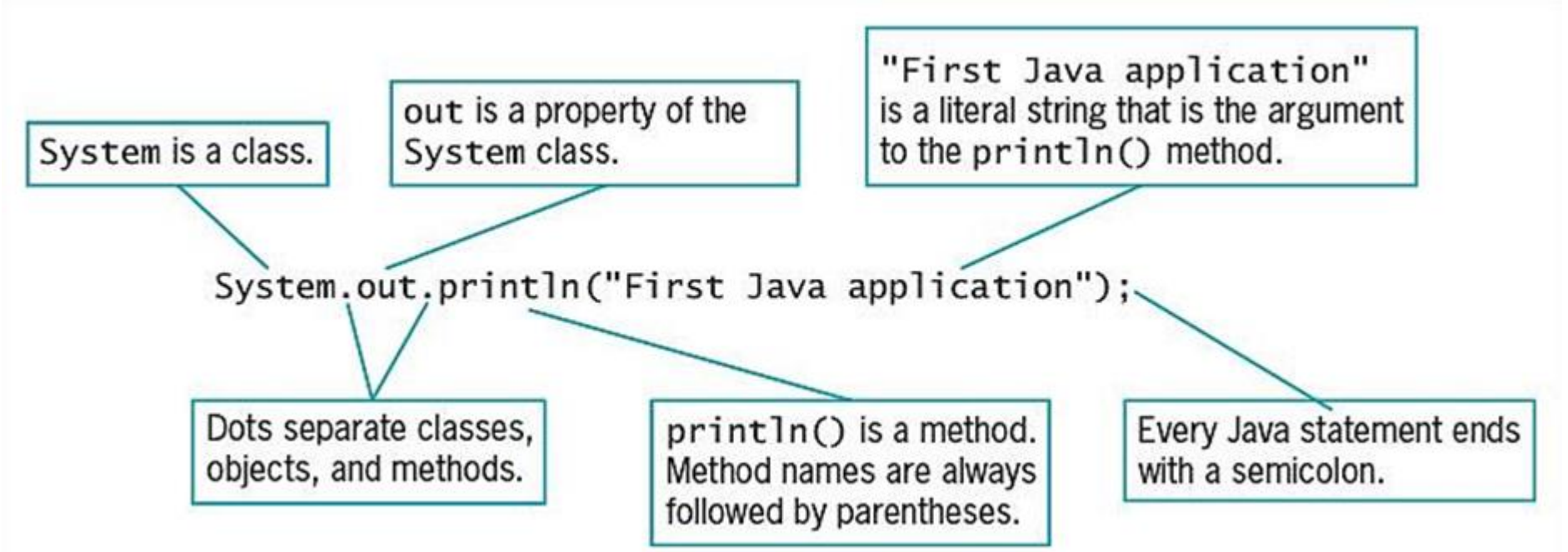
HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- A statement is a line of code ending in a semi-colon
- Statements inside the main method get executed in order
- The one statement in this program prints a message to the console

Output Statement

- Use
  - System.out.print("First Java application"); or
  - System.out.println("First Java application");
- Anatomy of a Java Statement



Reserved words

- Also called keywords
- These words mean something specific to the compiler
- Colored or **bolded** when you type them in SciTE and most Java IDEs

Example

The word *class* is a reserved word. The compiler knows the word after *class* is the name of the class.

- There is a list of Java keywords in Appendix A of the textbook.

Java Reserved Words

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

A simple Java program: Blocks

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

- The outer pair of curly braces enclose everything in the HelloWorld class
- The inner pair of curly braces enclose all the statements in the main method
- A pair of curly braces along with everything inside is called a block

Special characters

– The ones we have seen in our HelloWorld program are:

Char	Name	Meaning
{ }	Curly braces	Denote a block of statements
[ ]	Brackets	Denote an array
( )	Parentheses	Used with methods
//	Double slashes	Precede a one-line comment
" "	Quotes	Enclose sequence of characters (message)
;	Semicolon	Marks the end of a statement

Anatomy of a Java Program

- Comments
- Reserved words
- Modifiers
- Statements
- Blocks
- Classes
- Methods
- The main method

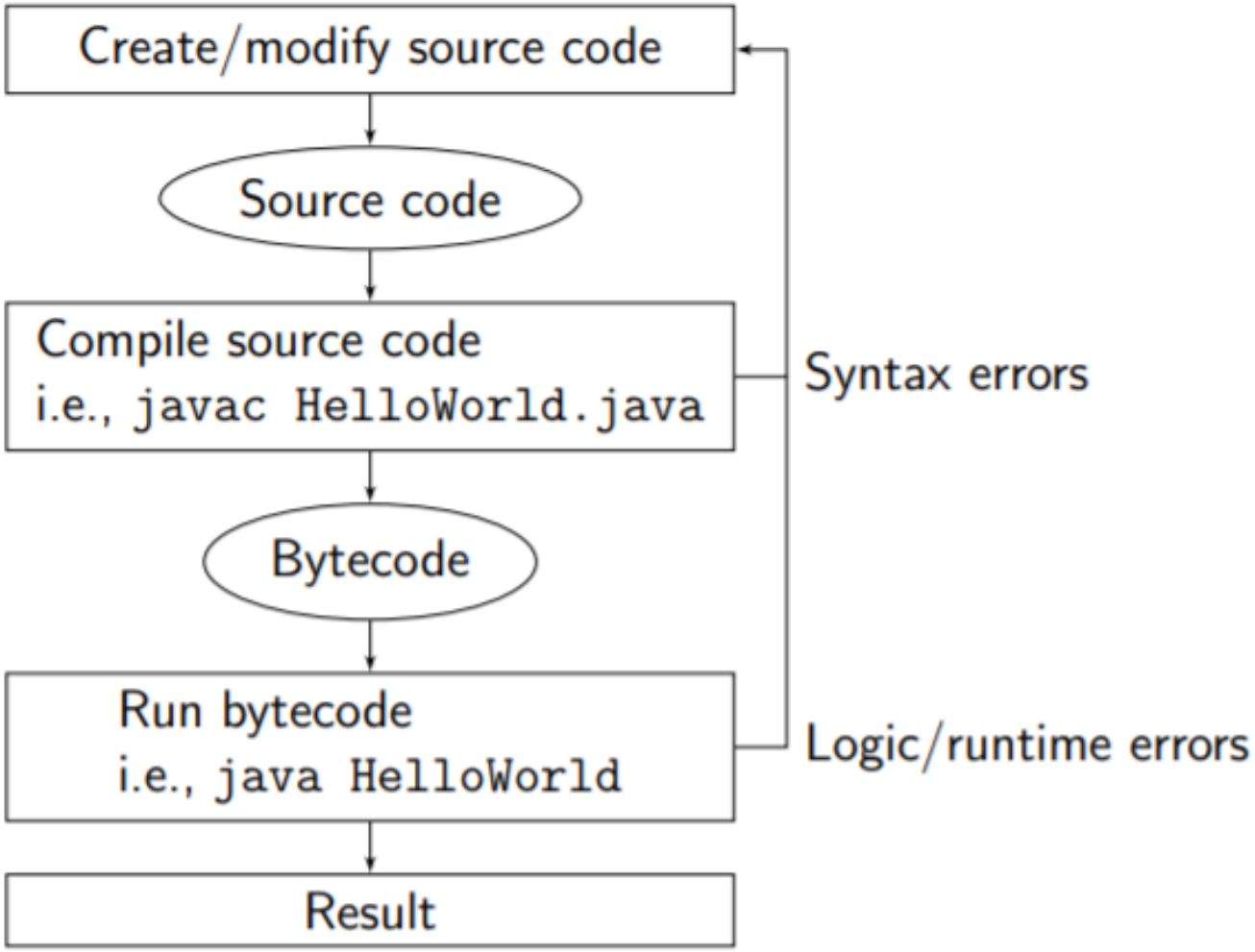
Example - Anatomy of a Java Program

- Let's analyse the following program and find out the role of each token (basic component of source code)

HelloWorld.java

```
public class HelloWorld{
  public static void main(String[] args){
    //Display the message Hello world! on the console
    System.out.println("Hello world!");
  }
}
```

Creating, compiling, and running a Java program



Compiling a .java file from the command line

- Open Command Prompt
  - Type cmd into your Windows search bar or in the Run dialog
- Navigate to the directory where your file is saved.
  - Type cd .. to move up a directory, and cd dirName to move down into a directory called dirName.
  - Alternatively, just type the entire path, i.e.,
  - cd c:\Users\YourName\dirName
- Type javac ClassName.java
  - Source code compiles into a .class file called
  - ClassName.class
  - Class file stored in same directory

Executing a Java program from the command line

- Open Command Prompt
  - Navigate to the directory where your .class file is saved
  - Hint: If you just compiled the program, you will already be in the correct directory
- Type java ClassName
  - Even though what you are running is the .class file, make sure not to include the file extension here
  - Program executes

Example

Let us see what it looks like to create, compile, and execute the HelloWorld program using the command line.

Three types of programming errors

Compiling and executing don't always go as planned.

- Syntax errors (compile error)
  - Detected by the compiler
  - Cause compiling to fail
- Exceptions
  - Cause the program to terminate abnormally
  - Happen in runtime (runtime error)
- Logic errors
  - Cause the program to produce the wrong output
  - Happen in runtime (runtime error)

Syntax errors

- Cause compile to fail
- Also referred to as compile errors
- Some usual causes:
  - Misspelled keywords
  - Missing punctuation
  - Unmatched braces, parentheses, quotes. . .
- Source code will not compile until these are fixed
- Many IDEs will indicate possible syntax errors as you type (before compiling)

Syntax error example

ShowSyntaxErrors.java

```
public class ShowSyntaxErrors {
  public static void main(String[] args) {
                System.out.println("Welcome to Java);
  }
}
```

```
>javac ShowSyntaxErrors.java
ShowSyntaxErrors.java:3: error: unclosed string literal
            System.out.println("Welcome to Java);
                               ^
ShowSyntaxErrors.java:3: error: ';' expected
            System.out.println("Welcome to Java);
                                              ^
ShowSyntaxErrors.java:5: error: reached end of file while parsing
}
 ^
3 errors
>Exit code: 1
```

Runtime errors

- Cause the program to terminate abnormally
- Some usual causes:
  - Attempting to access memory that is out of bounds
  - Dividing by zero
- Source code will compile
- Program will not fully execute until these are fixed

Example – exception

Code including the following statement causes a runtime error:

```
System.out.println(1/0);
```

Logic errors

- Cause the program to have unintended behavior, i.e., produce the wrong output
- Code with a logic error isn't "wrong"
  - Source code will compile
  - Program will execute
  - However. . . the result may be surprising
  - You've simply solved a different problem than the one you are trying to solve
- Most difficult type of error to find

More Practice

- Work on the lab of the week!