CS 302: Introduction to Programming

Lectures 2-3
Review

- What is a computer?
- What is a computer program?
- Why do we have high-level programming languages?
- How does a high-level program get translated into machine-readable instructions?
Review

- What is an IDE?

- What is the JVM?

- What is a method? What is the only method we will be dealing with for now?
Review

• Every Java statement must end with what?

• What is a String?

• How do you print out a String?

• What do squiggly-braces do? (these things: { })
A problem has been detected and Windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

WXYZ.SYS - Address F73120AE base at C0000000, DateStamp 36b072a3

Kernel Debugger Using: COM2 (Port 0x2f8, Baud Rate 19200)
Beginning dump of physical memory
Physical memory dump complete. Contact your system administrator or technical support group.
Errors

- 2 types:
  - Compile-time
    - Commonly syntax errors
    - You will not be able to run your program
    - Example: Forgetting a semicolon at the end of a statement, having braces ( { } ) that don't align, etc.
  - Run-time
    - Logic errors, exceptions
    - You WILL be able to compile and run your program, however it will not produce the results you expected
    - Example: cheats/glitches in video games, BSOD
    - Harder to catch
Storing Data

- Data stored in variables
- Many types of variables in Java
- Every variable has:
  - Type
  - Name
  - Value
  - Memory Location
- Can think of a variable as a storage crate or a parking space in a garage
- 2 steps to using variables
  - Declaration
  - Initialization
Declaration and Initialization

- Both steps must be done before a variable can be used
- Declaration must be done before initialization
- Declaration:
  - [type] variableName
- Initialization:
  - variableName = [initial value]
- Can be combined into 1 step:
  - [type] variableName = [initial value]
Common Variable Types

- Each variable type represents a different type of data

- **int:**
  - Short for integer
  - Stores whole numbers
  - Ex. 7, 100, -5, 0

- **double:**
  - Stores floating-point numbers (decimals)
  - Ex. 5.123, -800.2, 0.23456, 3.14159, 8.0
Common Variable Types

- Each variable type represents a different type of data

**Strings:**
- Stores sequences of characters
- Delimited by double quotes (" ")
- Ex. “Hello, how are you?”, “blah blah blah!”

**char:**
- Stores a single character
- Delimited by single quotes (' ')
- Ex. 'a', 'z', '!', '1'
WHAT ARE YOU DOING?
POSTING HOMEWORK ANSWERS ON THE INTERNET.

WHAT FOR?
UNLIKE HOLLYWOOD AND THE RECORDING INDUSTRY, I VIEW ONLINE COPYING AS A GOOD THING.

ISN'T IT GREAT?!

THIS IS SO UNLIKE YOU, JASON.

HOW SO?

SOON, THROUGH THE MAGIC OF PEER-TO-PEER FILE SHARING, KIDS THE WORLD OVER WILL HAVE READY-TO-USE SOLUTIONS TO EVERY QUESTION IN MY TEXTBOOKS! AND THERE'S NOT A THING TEACHERS CAN DO TO STOP IT.

USUALLY YOU'RE FREAKISHLY COMPETITIVE ABOUT YOUR GRADES.
YOU MISUNDERSTAND— THESE AREN'T THE CORRECT ANSWERS.
Initialization and Declaration Revisited

- **int x;**
  - Declaration: [type] variableName;
- **x = 10;**
  - Initialization: variableName = [initial value]
- **int x = 10;**
  - Declaration and Initialization combined

- **What's wrong with:**
  - int x = 5.2;
Label each of the following as initialization or declaration or combined:

- String name = “Bob Smith”;
- int myInt;
- double pi;
- char myChar = 'q';
- pi = 3.1415936;
- myInt = 0;
Variable Names

• Rules:
  – Must start with a letter or underscore (_)
  – Can include (but not start with) numbers
  – Cannot use spaces or other symbols like ? or %
  – Case sensitive (int x is different from int X)
  – Cannot use reserved words (i.e. double double or int public)

• Conventions
  – camelCase: start with lower case, first letter of each additional word is capitalized
Break 2
Working with Variables

• Assignment operator: =
  – Means “sets”
  – [destination] = [input value];
  – int x = 5; → x is set to have the value 5
  – x = 6; → x's value is now set to 6

• Can assign variables to other variables
  – int x = 5; → x is set to 5
  – int y = x; → y is set to 5
  – x = 6; → now x is 6, y is 5
Displaying Variables

- Use `System.out.println([variableName]);`
  - No double quotes around variable name

```java
int x = 5;
System.out.println(x); → prints out: 5
vs.
System.out.println(“x”); → prints out: x
```
Displaying Variables and Strings

- Use `System.out.println(“[String]” + [variableName] + “[String]”);`

```java
int x = 5;
System.out.println(“My number is: “ + x);
```

**Output: ** *My number is: 5*

```java
System.out.println(x + “ is my number”);
```

**Output: ** *5 is my number*

```java
System.out.println(“My number is: “ + x + 2);
```

**What is the output?**
Arithmetic

- Addition: +, ++
- Subtraction: -, --
- Multiplication: *
- Division: /
  - Be careful with integer division
- Expressions:
  - variableName = [variable1] [operator] [variable2]
  - int x = 2 + 2;
  - double y = 4.5 / 3;
  - int a = x * 2;
Parenthesis

Method calls (i.e. `System.out.println();`)

Order of Operations

```java
int x = 3 + 4 * 7; x is assigned 31
int x = (3 + 4) * 7; x is assigned 49
int x = (3 + 4) * 7 - 2; x is assigned 47
int x = ((3 + 4) * 7) - 2; x is assigned 47
```

Can mix

```java
System.out.println((3 + 4) * 7);
```

Must be balanced

```java
System.out.println((3 + 4 * 7);
```
Mixing Integer and Floating-Point Operations

Fine as long as we are assigning back to a double

```c
int x = 4, y = 2;
double z = 5.5;
z = (z + x) * y;
z is assigned (5.5 + 4) * 2 = 19;
```

Could we have done `x = (z + x) * y;` instead?
Integer Division

- Deceptively simple
- Do division and cut off remainder (DO NOT ROUND)

```java
int x = 2 / 2; x is assigned 1
x = 5 / 2; x is assigned 2
x = 1 / 2; x is assigned 0
```
Modulus Operator - %

- Mostly used with ints though it can be used with doubles
- Gets the remainder after integer division

```java
int x = 2 % 2;
- x is assigned 0 because 2 / 2 = 1 remainder 0

x = 5 % 2;
- x is assigned 1 because 5 / 2 = 2 remainder 1

x = 1 % 2;
- x is assigned 1 because 1 / 2 = 0 remainder 1
```
Powers and Roots

- No symbol for powers or square roots on a keyboard*
- Must call pre-built methods in the Math library

Math.pow([base], [power]);

- double d = Math.pow(2, 3); d is assigned 8 because $2^3$ is 8
- d = Math.pow(3, 2); d is assigned 9 because $3^2$ is 9

Math.sqrt([number]);

- d = Math.sqrt(4);
- d = Math.sqrt(-1);

These methods return doubles

*Often times the ^ will be used for powers e.g. $2^3 = 8$
Method Calling Conventions

- `[identifier].[methodName]([arguments])`
- `System.out.println("Hello");`
- `Math.pow(2, 4);`
- `System.out.println(x);`
- Multiple **arguments** always separated by commas (,)
- **Identifiers** can be class or variable names
Break 3

- Write a program that stores values for a, b, and c and solves the quadratic equation

\[ ax^2 + bx + c = 0 \]

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]
Comments

- Enables writing notes in your code
- Compiler ignores
  
  // - single line

  /*...*/ - multi-line

- Javadoc - /**...*/ - multi-line and converts to html
- Best practice – USE these
- Difficult to over-comment
- See style pages on website after it is created
Constants

- What if we have a variable we know will never change? (Gravitational Constant = ?)
- Could just write in the literal value every time we need it
- Better practice – use constants
- Specified by the "final" keyword
  - final double MILES_TO_KM = 1.609344;
- Naming conventions – ALL_UPPER_CASE
- Avoid magic numbers – use comments
Input

- Prompt – when the computer asks the user for input and waits
- Using the Scanner object:

```java
import java.util.Scanner; //above the main method
...

//in the main method
Scanner in = new Scanner(System.in);
System.out.println("Enter an int:");
int x = in.nextInt(); //x gets the value of whatever the user enters
```
Common Scanner Methods

- **next()**
  - Read in up to a space or newline
  - Returns a String
- **nextLine()**
  - Reads in an entire line
  - Returns a String
- **nextInt()**
  - Reads in and returns an int
- **nextDouble()**
  - Reads in and returns a double
Write a program that:

- Outputs: *What is your name?*
- Waits for the user to enter their name
- Outputs: *I'm sorry [name], I'm afraid I can't do that...*
Primitive vs Reference Types

- Primitives
  - int, double, char, boolean, byte, long, short, float
- Reference
  - String and other user-defined types
- Primitives hold a literal value
  - Can't call methods on them
- References hold a memory location where that value is stored
  - Can often call methods on them
Casting

- What if I wanted an int back from a Math.pow or Math.sqrt call?
  - `int x = (int) Math.pow(2,3);`
    - `x` is assigned 8
    - How is this different than double `y = Math.pow(2,3)`?
  - Be careful – lose all fractional part (DOES NOT ROUND)

```java
double y = Math.sqrt(5);
int x = (int) y;
x is assigned 2
```
Converting Strings to Numbers

- **String → int**
  - `Integer.parseInt(String)`
  - `String aNumber = "5";
  - `int x = Integer.parseInt(aNumber);`

- **String → double**
  - `Double.parseDouble(String)`
  - `double y = Double.parseDouble("2.2");`
Print vs PrintLine

- System.out.print([stuff to print out]);
  - Prints String with no newline on the end
- System.out.println([stuff to print out]);
  - Prints String with newline added on the end

System.out.print("This is a print");
System.out.println("This is a println");
System.out.println("Another println");

This is a print
This is a println
Another println
Practice

- Write a program to compute the volume and surface area of a cylinder
  - Prompt the user for height and diameter
  - \( V = \pi r^2 \times \text{height} \)
  - \( SA = 2 \times \pi r \times \text{height} \)
  - Print out the volume and surface area
Using Objects

3 Steps

- Import the package
  - import java.util.Scanner;
  - import java.util.Random;

- Instantiate a new Object
  - Scanner in = new Scanner(System.in);
  - Random randGenerator = new Random();

- Call methods on the Object
  - in.next();
  - randGenerator.nextInt();
How to Generate a Random Number?
The Random Object

● What is a seed?
  ● A starting value for the random number generator that enables it to generate the same sequence of numbers each time
  ● If no seed is passed, Java uses the system clock as a seed value meaning every time the program is run it starts with a different seed value (since the time is different) which gives appearance of randomness
The Random Object

- 2 ways to instantiate
  - Random randGenerator = new Random();
  - Random randGenerator = new Random(seed);
    - Seed is of type long

- What is a long?
  - Another primitive type – looks like an int but can hold more values
  - ints = -2,147,483,648 – 2,147,483,647
  - Java can automatically convert ints to longs for you
Working with the Random Object

- It's an object -> call methods!

```java
Random rand = new Random();
int x = rand.nextInt(); // x is any int value
int y = rand.nextInt(10); // y is 0-9
```

// How to get a number 1-10?
Break 5

- Write the code to simulate a D20 (20-sided dice)
  - Input: Nothing (or a seed if you want...)
  - Output: Random # from 1 – 20 inclusive