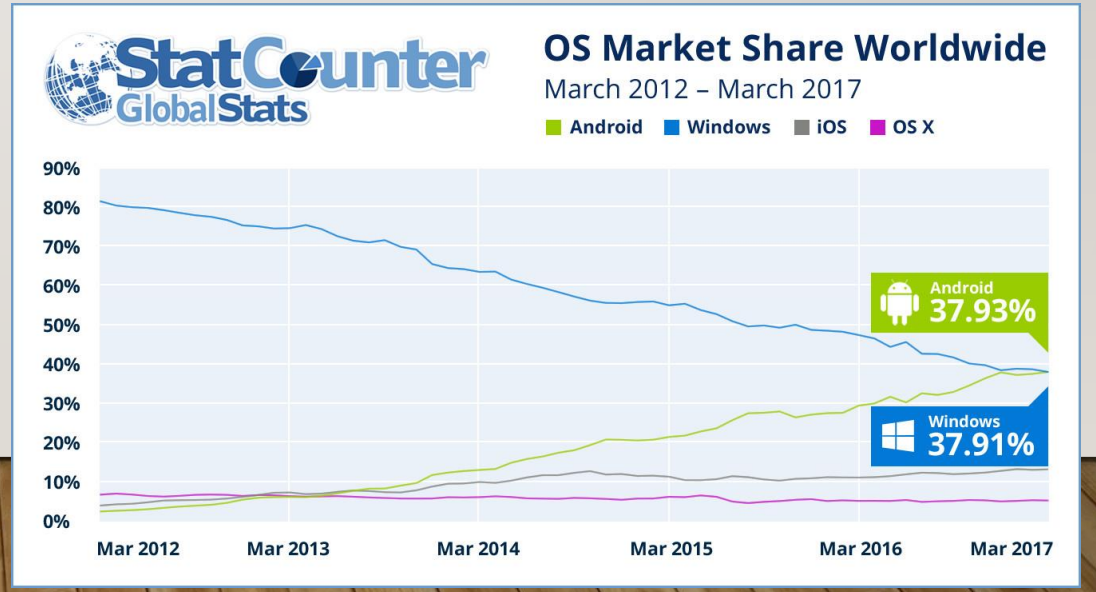
GETTING STARTED WITH INTELLIJ AND HOW JAVA WORKS

CS 3140 – LECTURE 02

JAVA

- Why do we use Java?
  - First: because it was covered in DSA1, I can be reasonably sure everyone in this room knows how to use it
  - Second: Java is a portable language
    - Portable – you can take your applications with you to different operating systems
  - Third: Android is now the most widely used operating system in the world today. It primarily uses Java for development.



UNDERSTANDING JAVA

- Computers cannot run .java files!
  - Instead, we compile Java files into .class files
- Eclipse and IntelliJ aren't Java!
  - These are IDEs used to assist programmers in writing Java programs

PORTABILITY EXPLAINED

- In many languages, it's normal to see something like this
- In Java, you wont' see this!

```
#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__NT__)
   //define something for Windows (32-bit and 64-bit, this part is common)
   #ifdef _WIN64
      //define something for Windows (64-bit only)
   #else
      //define something for Windows (32-bit only)
   #endif
#elif __APPLE__
    #include <TargetConditionals.h>
    #if TARGET_IPHONE_SIMULATOR
         // iOS Simulator
    #elif TARGET_OS_IPHONE
        // iOS device
    #elif TARGET_OS_MAC
        // Other kinds of Mac OS
    #else
    #   error "Unknown Apple platform"
    #endif
#elif __linux__
    // linux
#elif __unix__ // all unices not caught above
    // Unix
#elif defined(_POSIX_VERSION)
    // POSIX
#else
#   error "Unknown compiler"
#endif
```

UNDERSTANDING JAVA

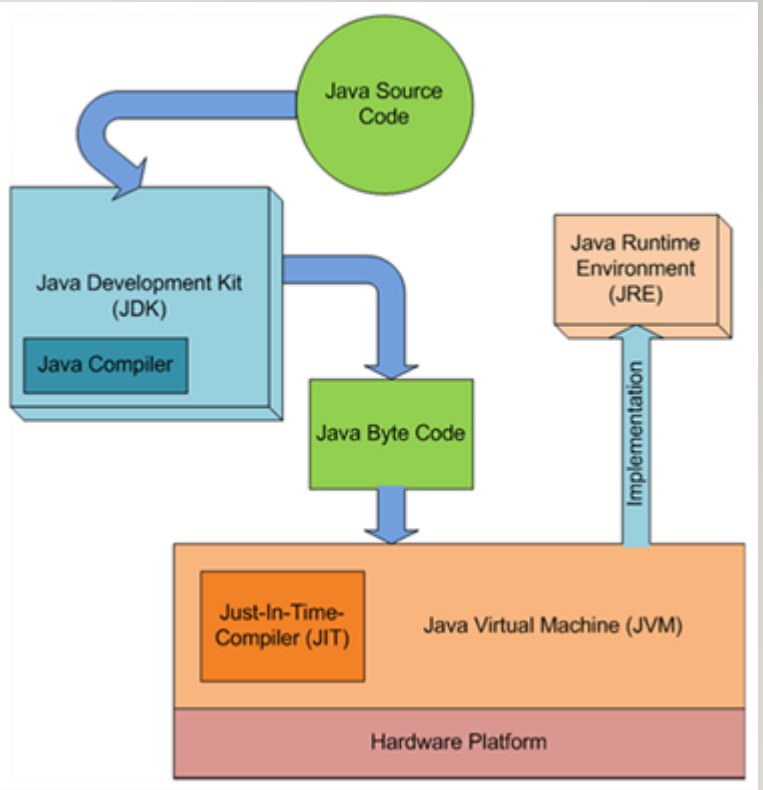
- All programming languages work by utilizing resources from the computer
  - Processor
  - Memory
  - Disc space
  - Monitor
  - Network Connection
- Most languages do this by directly interfacing with the operating system
  - This means, however, that a program that expects a file to be stored in a Windows way won't work of a Mac.
  - Different memory architectures could break the program

JAVA COMPILER

- The Java compiler effectively takes in a .java file and outputs a .class file
- The .java file is human readable source code
- The .class file is not human readable, but lists the machine instructions (for the JVM*) to run the Java program

UNDERSTANDING JAVA

- JRE – Java Runtime environment
  - An abstract computing machine that is used to execute .class files.
- JVM – Java Virtual Machine
  - An implementation of a JRE, which interfaces with the underlying operating system, hardware, etc.
- JIT – Just In Time compiler
  - JIT is part of the JVM – translates instruction sets of the JVM to instruction sets of the CPU



INTELLIJ VS. ECLIPSE

- In this class, I will recommend using IntelliJ for all homeworks, lecture material, etc.
  - I will use IntelliJ in class, and slides will use IntelliJ screenshots
  - Why not Eclipse?
    - Git tends to work much better in IntelliJ in my experience
    - Gradle doesn't play very nicely with Eclipse: you can make it work but it's a pain
    - Getting JavaFX to work in IntelliJ is fairly simple; in Eclipse, it's a nightmare
      - It took me several hours to figure out how to do it, and I ended up with 3 pages of instructions
- If your used to Eclipse, IntelliJ is VERY similar
  - Same developers as PyCharm (CS 1110/1111)

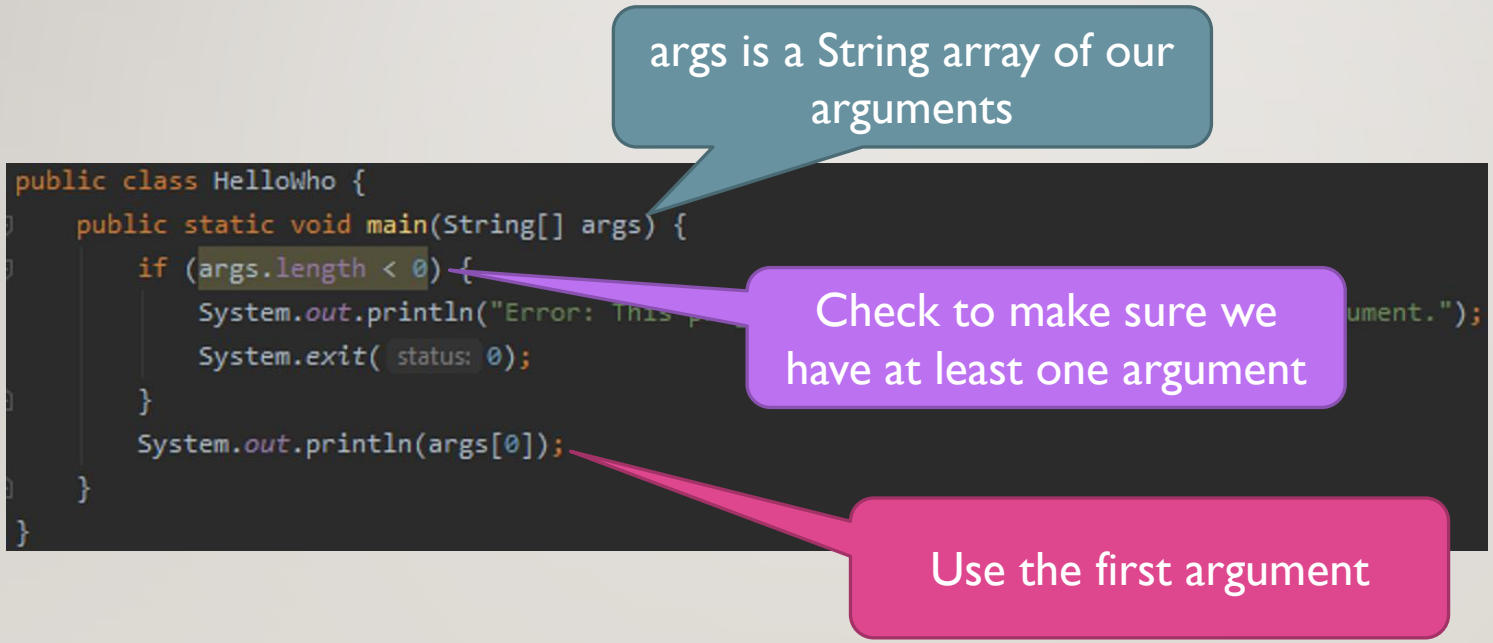
COMMAND LINE ARGUMENTS

- Let's start by writing a simple program that uses Command Line Arguments

```
public class HelloWho {
    public static void main(String[] args) {
        if (args.length < 0) {
            System.out.println("Error: This program needs at least one command line argument.");
            System.exit( status: 0);
        }
        System.out.println(args[0]);
    }
}
```

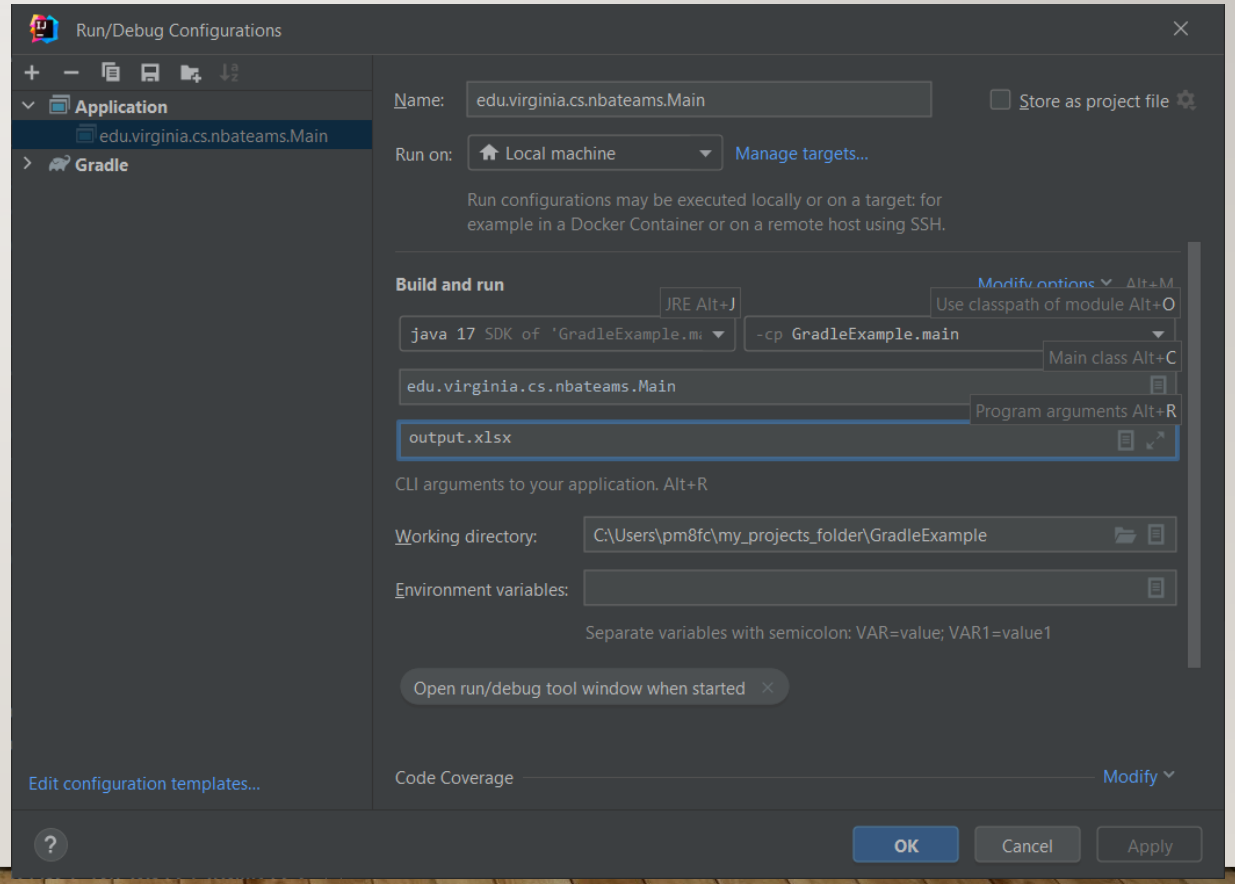
COMMAND LINE ARGUMENTS

- Let's start by writing a simple program that uses Command Line Arguments



SETTING ARGUMENTS IN INTELLIJ

- Run->Edit Configurations…
- Ensure Main Class is the class you want to run
  - Under "Main Class"
  - Must include package name
- Add Arguments to Program Arguments
  - You can create multiple configurations to test with different arguments



EXAMPLE ARGUMENTS

- `Steve 15 "Go Hoos"`
- If these were your program arguments, your args array would be:
  - args[0] – "Steve"
  - args[1] – "15" – String 15, not int 15!
  - args[2] – "Go hoos"

- `args.length` – the number of command line arguments

THIS SEEMS AN INEFFICIENT WAY TO ADD INFORMATION…

- Why don't we just hardcode things?

THIS SEEMS AN INEFFICIENT WAY TO ADD INFORMATION…

- Why don't we just hardcode things?
  - Because we won't always run the code from IDE

RUNNING JAVA PROGRAMS WITHOUT AN IDE

- Open a terminal
  - You can also open a terminal in your IDE (click terminal tab at the bottom) or use an app like Terminal (Mac/Linux) or Powershell (Windows)
- Navigate to your project's Build director, and find HelloWho.class
  - Likely in …\build\classes\java\main
- To run this class, type:

```
java HelloWho Steve
```

```
PS C:\Users\pm8fc\IdeaProjects\Temp\build\classes\java\main> java HelloWho Steve
Hello, Steve
PS C:\Users\pm8fc\IdeaProjects\Temp\build\classes\java\main> java HelloWho James
Hello, James
PS C:\Users\pm8fc\IdeaProjects\Temp\build\classes\java\main> java HelloWho "Simon Paul" "Art Garfunkle"
Hello, Simon Paul
PS C:\Users\pm8fc\IdeaProjects\Temp\build\classes\java\main>
```

RUNNING JAVA PROGRAMS WITHOUT AN IDE

- We now have a means to easily run Java programs without relying on an IDE
  - This can be done by a human
  - Done by a server
  - Run by another Java program, or a program in any language!
- The .class file can be seen as a distributable class
  - That is, I can upload the .class file to the internet, and share the runnable program *without* sharing the source code
- Bundle a group of class files (and other files) together into a .jar for even more ease!

MAKING JAR FILES

- A JAR file is like a bundle of class files. The JAR file can be configured with a Manifest to be runnable
- Run a .jar file with:

```
java –jar NameOfTheJarFile.jar
```

- This allows you to hide the structure of the source code, and instead just release a runnable application.
- In IntelliJ, we will do this via Gradle (one week from now)

RELEASE THE EXECUTABLE, NOT THE SOURCE CODE

- In patenting and licensing, results are not protected, only the means to retain those results
- As this relates to software
  - Source code can be patented and protected
  - What the source code does cannot be
- Further, companies go to great lengths to protect source code

Guy Finds *StarCraft* Source Code And Returns It To Blizzard, Gets Free Trip To BlizzCon

Nathan Grayson
5/03/17 7:00PM • Filed to: STARCRAFT

970.6K 401 87

Last month, Reddit user Khemist49 made a truly unlikely find: a gold master source code disc of the original *StarCraft*. From 1998. At first, he didn't know what to do with it. Ultimately, he sent it to Blizzard, who was very grateful to

Popular Deals on The

Thursday's Amazon W More

Tercius, Corey

Amazon W Off Alread

Tercius

THINGS FOUND IN SOURCE CODE

```
//
// Dear maintainer:
//
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 42
//
```

```
Exception up = new Exception("Something is really wrong.");
throw up;  //ha ha
```

```
public abstract class RichardIsAF[redacted]IdiotControl : MobileBaseControl, ICompanyProfileControl
{
    protected abstract Pager Pager { get; }

    public void BindCompany(int companyId) { }

    public RichardIsAF[redacted]IdiotControl()
    {
        MakeSureNobodyAccidentallyGetsBittenByRichardsStupidity();
    }

    private void MakeSureNobodyAccidentallyGetsBittenByRichardsStupidity()
    {
        // Make sure nobody is actually using that f[redacted] bindcompany method
        MethodInfo m = this.GetType().GetMethod("BindCompany", BindingFlags.DeclaredOnly |
            BindingFlags.Instance | BindingFlags.Public | BindingFlags.NonPublic);
        if (m != null)
        {
            throw new RichardIsAF[redacted]IdiotException("No!! Don't use the f[redacted] BindCompany method!!!");
        }
        // P.S. this method is a joke ... the rest of the class is f[redacted] serious
    }

    /// <summary>
    /// This returns true if this control is supposed to be doing anything
    /// at all for this request. Richard thought it was a good idea to load
    /// the entire website during every request and have things turn themselves
    /// off. He also thought bandanas and aviator sunglasses were "f[redacted]
    /// gnarly, dude."
    /// </summary>
    protected bool IsThisTheRightPageImNotSureBecauseRichardIsDumb()
    {
        return Request.QueryString["Section"] == this.MenuItemKey;
    }
```

JAR MANIFEST

- A Jar file is *very* similar in concept to a zip file
  - It contains several files and folders
  - Typically, a jar contains:
    - The .class files (stored in a matching project structure)
    - Any resource files
  - Also contains a Manifest (typically stored as "META_INF/MANIFEST.MF"
  - Manifest tells Java relevant information about the Jar file
    - For us, the one we are most interested in is Main-Class
    - Tells Java which class to run when Jar is executed
    - That class must have a public static void main(String[ ]) method