



Introduction to Java

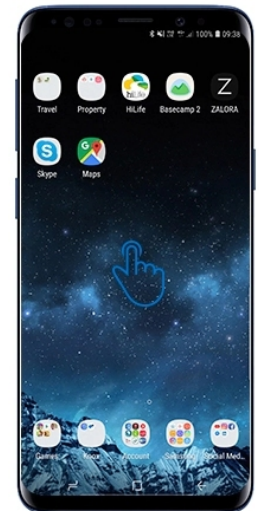
James Brucker

What is Java?

Java is a language for writing computer programs.

✓ *it runs on almost **any** "computer".*

- desktop
- mobile phones
- game box
- web server & web apps
- smart devices



What Uses Java?

❑ OpenOffice & LibreOffice (like Microsoft Office)

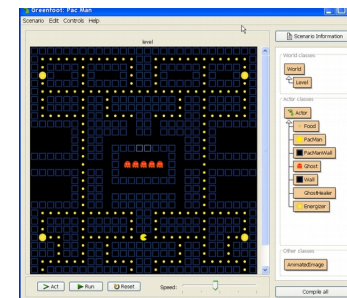
❑ Firefox

❑ Google App Engine

❑ Android

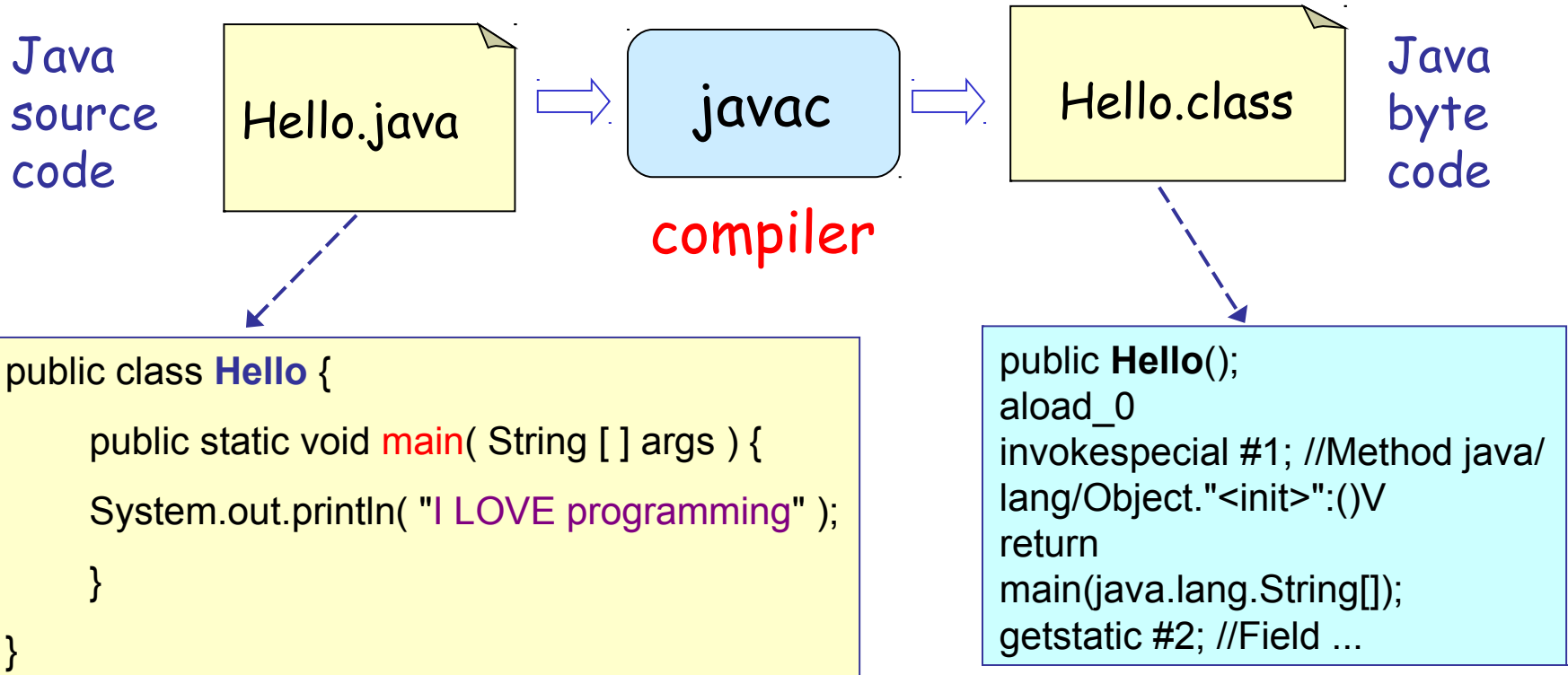
❑ MindCraft & Greenfoot

❑ IDE: Eclipse, IntelliJ, NetBeans



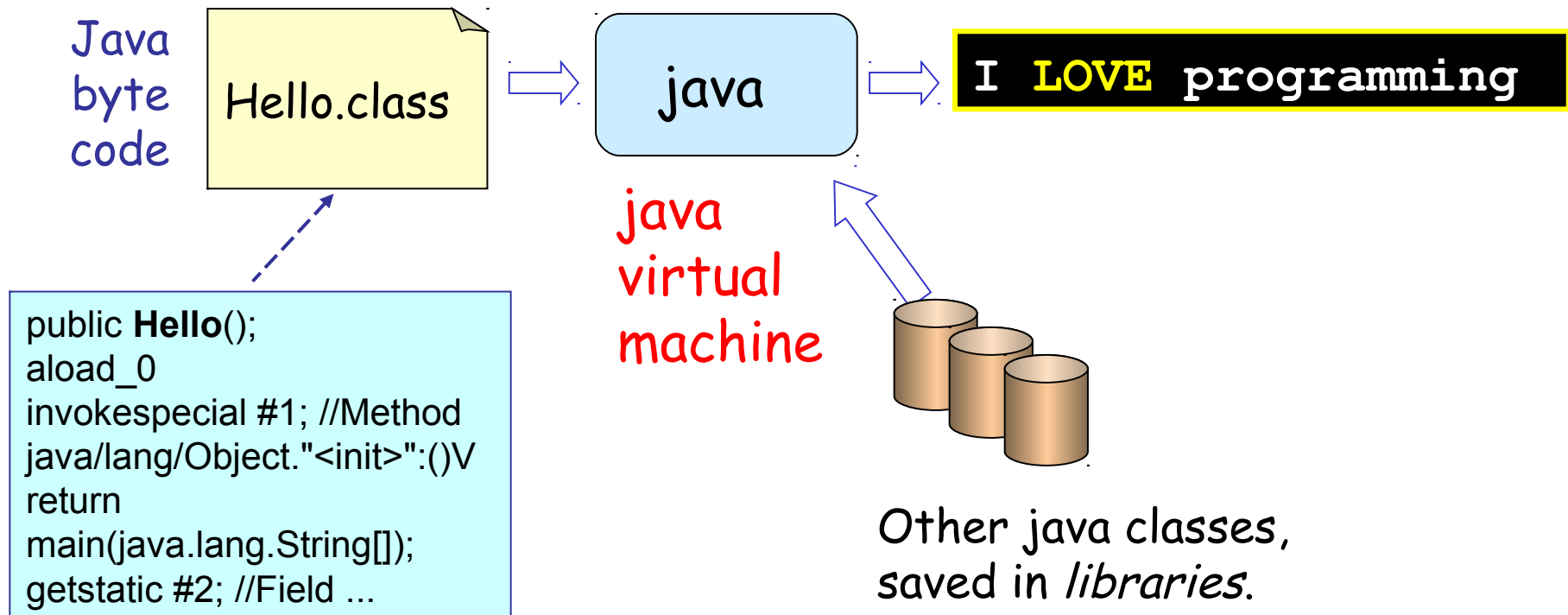
How Java Works: *Compiling*

1. You (programmer) write **source code** in Java.
2. A **compiler** (`javac`) checks the code and *translates* it into "**byte code**" for a "virtual machine".



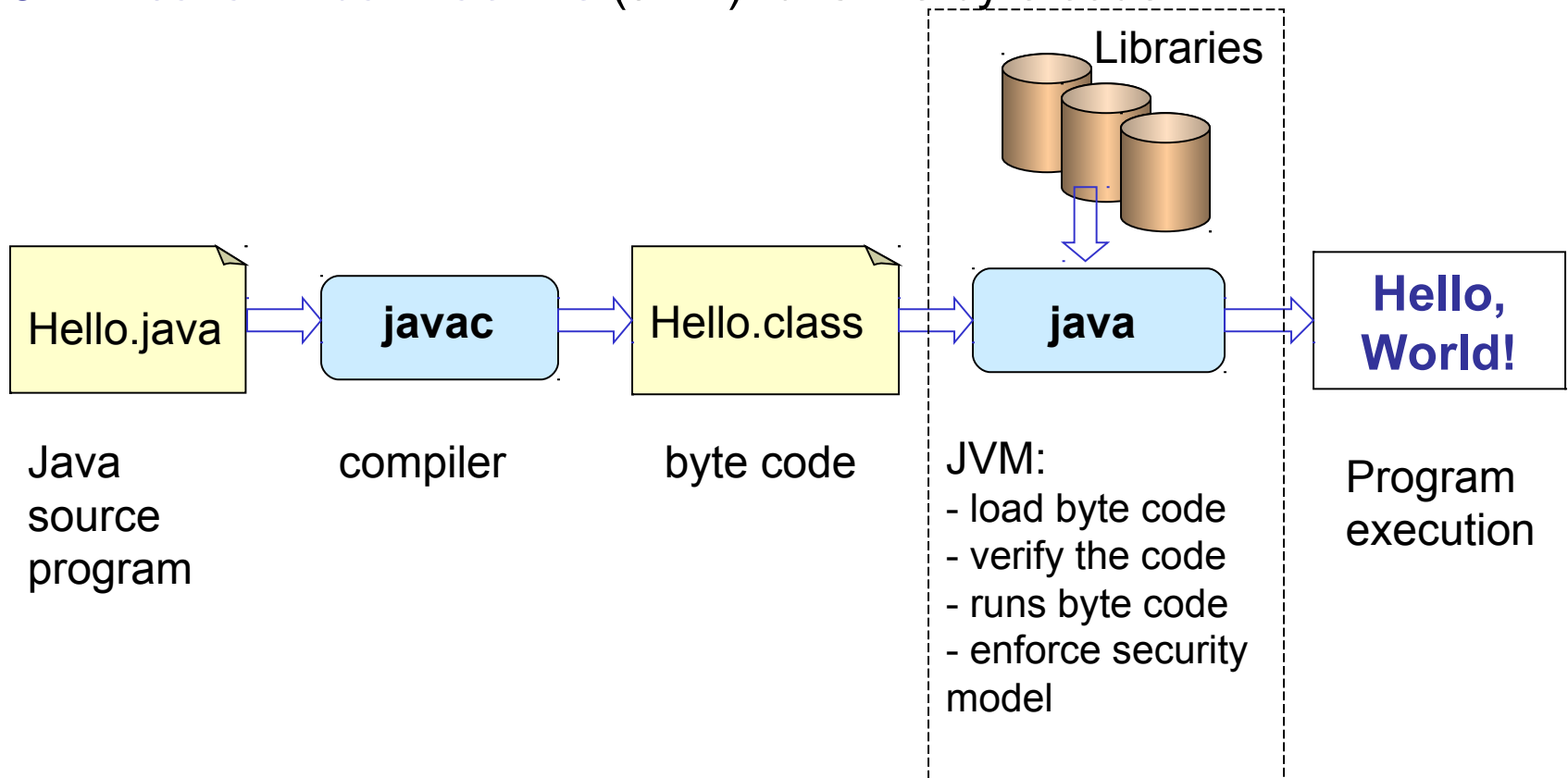
How Java Works: *Run the Bytecode*

3. *Run* the byte code in a Java Virtual Machine (JVM).
4. The JVM (java command) *interprets* the byte code and loads other code from libraries as needed.



Summary: How Java Works

1. You (programmer) write program source code in Java.
2. Java compiler checks the code and translates it into "byte code".
3. A Java virtual machine (JVM) runs the byte code.



How to Write Source Code?

You can use *any editor* to write the Java source code.

For your first program, use a **text editor** (e.g. Notepad).

Later you will use an **Integrated Development Environment** (IDE) such as Eclipse, IntelliJ, Netbeans, or VS Code. Or a beginner's IDE: BlueJ.

An IDE makes you much more productive, but you need some practice... so for now, use a text editor.

How to Get Java

To do these examples you must install the Java Development Kit (JDK).

Useful to also install the Java documentation (Javadoc).

Implementations of Java

- ✓ Oracle Java: java.oracle.com
- ✓ OpenJDK: AdoptOpenJdk.net
- ✓ Stick with LTS versions: Java 8, 11, 15

Install Java Development Kit (JDK)

Your Choice:

1. Oracle Java

<https://www.oracle.com/technetwork/java/javase/downloads/>

2. OpenJDK Java (with choice of JVM)

<https://adoptopenjdk.net/>

Installation Notes

- ❑ There are some suggestions at the end of these slides
- ❑ You can also find videos on YouTube for help installing Java and writing your first program.

Check your Java Installation

1. Open a command line window or "terminal" (Mac). This is where you will enter commands.

```
C:>
```

2. Enter "java -version" and "javac -version"

```
C:> java -version
```

```
java version "11.0.5" 2019-10-15 LTS
```

If it prints "command not found", then the Java "bin" directory is not on your terminal's "search path", specified by the **PATH** environment variable.

What Are We Going to Write?

First version: program prints a greeting

```
Hello, human.
```

Second version: program will ask for your name, save it as a String, and print a personal greeting. Example:

```
What is your name? Cat
```

```
Hello, Cat.
```

Third version: we will get the current time and use it to print a time-aware greeting. We will also write a method for this.

```
Good morning, Cat. (before 12:00)
```

```
Good afternoon, Cat. (after 12:00)
```

Write this Program

1. Use a **text editor** (WordPad, EditPlus, *not* MS Word).
2. Save the file as **Hello.java**

Name of the class must be same as name of file

```
public class Hello {  
    public static void main( String[ ] args ) {  
        System.out.println( "Hello, human" );  
    }  
}
```

Compile and Run the Program

3. *Compile the program using "javac"*. This translates the source code into byte code.

```
C:> javac Hello.java
```

4. *Run it.* "java" runs the virtual machine.

```
C:> java Hello
```

```
Hello, human.
```

Errors?

javac: Command not found.

This means Java isn't on your PATH environment variable. Fix it. :-)

Hello.java:1:public Class Hello {

^ class, interface, or enum expected

Messages like this are syntax errors.

Check your typing. Java is case-sensitive.

The main method must be "main", *not* "Main".

In this example, the source code contains "Class" instead of "class".

2nd Version: read some input

In this version we'll see how to read input and use a local variable.

The easiest way to read input in Java is with a Scanner object. A Scanner *parses* input into strings, numbers, etc

To create a Scanner *object* for reading the console use:

```
import java.util.Scanner;
```

```
public class Hello {
```

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

```
        Scanner console = new Scanner( System.in );
```

`import` tells the compiler where to find the Scanner class

this declares a variable of type Scanner named "console".

System.in is a predefined variable for byte-by-byte input from the default input (the terminal)

2nd Version: main method

Scanner has a *method* named `nextLine()` that reads the rest of the input line and converts it to a `String`.

`System.out.print("string")` does not print a newline after the output.

```
public static void main(String [] args) {  
    Scanner console = new Scanner(System.in);  
    System.out.print("What is your name? ");  
    String who = console.nextLine( );  
    System.out.println("Hello, " + who );  
}
```

+ joins two strings together.


dot notation for object methods

`console` is the name of a variable (an *object reference*) that *refers* to a Scanner object.

To invoke an object's **method**, you write the object reference (variable) + "." + the method name:

```
String who = console.nextLine( );
```

nextLine method of Scanner object



Strings are also objects and have methods. To get the length of a String use:

```
int n = "hello world".length( );
```

Compile and Run 2nd Version

Compile (javac) and run (java) the 2nd version:

```
C:> javac Hello.java
```

```
C:> java Hello
```

```
What is your name? Nerd
```

```
Hello, Nerd.
```

3rd Version: decisions

Most programs contain *logic* for *decision making*.

We'll write a program that greets and shows the time.

The logic is:

```
get the current time
```

```
if now is before 12:00
```

```
    print "Good Morning, " + user's name
```

```
else if now is before 18:00 then
```

```
    print "Good Afternoon, " + user's name
```

```
else print "Good Evening, " + user's name
```

```
print "It is now hh:mm:ss" (show the actual time)
```

Creating a Date Object

Java has a `Date` class (also located in `java.util` package).

`Date` has *lots* of methods. `getHours()` returns the hour.

To create a `Date` *object* with the current date and time, use:

```
Date now = new Date( );
```

To create an *object* from a class, write:

```
new ClassName( )
```

Some classes allow parameters when creating a new object, for example:

```
new Scanner( System.in )
```

3rd version: using a method


```
import java.util.Date;
import java.util.Scanner;
/** Greeting with time of day. */
public class Hello {
```

add the greet method here

```
    public static void main( String[] args ) {
        Scanner console = ...;
        // ask user his name and call greet
        System.out.print("What is your name? ");
        String who = console.nextLine( );
        greet( who );
    }
}
```

3rd version: greet method

This declares a method named `greet` that doesn't return any value (`void`) and has one parameter (`name`) that is a `String`



```
public static void greet( String name ) {  
    Date now = new Date( );  
    if ( now.getHours() < 12 )  
        Sytem.out.println("Good morning, "+name);  
    else  
        System.out.println("Good afternoon, "+name);  
    System.out.println("The time is "+now);  
}
```

Compile and Run

- Compile (**javac**) and run (**java**) the 3rd Hello program.
- You will notice that it prints the date and time. If you want to print only the time, such as 12:45:00, use this:

```
System.out.printf("The time is now %tT\n",  
now );
```

printf creates *formatted output*. The first string is the format (including text to print). **%tT** is a *format code* to output a time value (using the next argument). You'll study formatted output later in the course.

It is similar to printf in the C language.

That's It!

- ❑ This program was a quick overview of how to program in Java.
- ❑ After this, we'll write code in BlueJ. BlueJ makes it much easier to edit, compile, and run your code.
- ❑ In BlueJ, you can also run Java commands interactively, without creating a program.

What do you need to use Java?

Programmer needs:

- Java compiler (javac)
- the Java foundation classes (included with JDK)
- documentation called "Java API" which can be viewed in a web browser

User who runs a program needs:

- Java Virtual Machine (JVM): java command
- foundation classes used by the JVM
- This is the "Java Runtime Environment" (JRE)

Where Are the Classes?

- ❑ The Java Runtime (also included in JDK) has **thousands** of classes.
- ❑ **Where are they?**
- ❑ Where is the Java compiler (**javac**)?



Know Your Tools

Craftsmen, Farmers, Engineers, Scientists, Artists ...

Know Their Tools

Layout of the JDK

C:/java

/jdk1.8.0_231

/bin

/demo

/include

/jre

/lib

/sample

src.zip

Location for java (your choice)

JDK 8u231

programs ("binaries")

demo apps with source

interface to C language code

Java Runtime for JDK

libraries used by JDK

samples with source

source code for JRE classes

Executable are in **/bin**

C:/java

/jdk1.8.0_231

/bin

java

Java Virtual Machine (JVM)

javac

compiler

javadoc

create Javadoc (HTML)

javaw

JVM for GUI applications

jar

manage Java Archive (jar) files

jvisualvm

monitor the Java VM

Demo applications

C:/java

/jdk1.8.0_231

/demo

/jfc/

/applets/

/plugins/

/netbeans/

"Demo" is optional.

You might not have it.

Demos for Java core classes

Java applets (run in browser)

Java plugins

jfc and applets as Netbeans projects

Java 11 JDK (new layout)

C:/java

/jdk11.0.5

/bin

/conf

/include

/jmods

/lib

/legal

release

Location for java (your choice)

JDK 11u05

programs ("binaries")

configuration files (.properties)

interface to C language code

Java module system

libraries used by JDK

legal notices for lawyers

version no. & release info

Java Archive (JAR)

A JAR file (* .jar) is a ZIP file with special directory structure.

- * Used to contain Java class files.
- * May also contain other kinds of files (images).
- * **You can run JAR files!** - if they designate a "main" class

Organize Your Software

You may have *many JDK and JRE*.
Organize them so you can find them.

C:/java

/jdk1.8.0_231

/docs

/jdk11u05

/docs

/tutorial

/...

Location for java (your choice)

JDK 8 update 231

Documentation for JDK 8

JDK 11 update 5

Javadoc for JDK 11

Java tutorials

More Java tools and libraries

Java on your PATH

```
JAVA_HOME=C:\java\jdk1.8.0_231
```

```
PATH=...;%JAVA_HOME%\bin;...
```

Java should be on your search PATH in order to be able to use Java at the command prompt.

May also affect double-click on executable JAR files.

Can't find **javac** or **java** ?

```
C:> javac -version  
"javac" not found.
```

If "Not Found" then add Java's "bin" dir to your PATH.

- 1) Right-click "My Computer" → choose "Properties"
- 2) Select "Advanced" tab
- 3) Click "Environment Variables"
- 4) Select PATH and click "Edit"
- 5) Add this to PATH:

```
...;C:\java\jdk1.6.0_26\bin
```

MS Windows

Env Variables used by Java

These are not required, but used by many Java tools to find Java. On Windows they are usually set by the setup program, but you might want to check it.

How? open a cmd window and type "set" or "env".

```
JAVA_HOME=C:\java\jdk1.8.0_231
```

```
JRE_HOME=C:\java\jdk1.8.0_231\jre
```

CLASSPATH is another variable used by Java.

CLASSPATH tells Java where to look for **class** and **jar** files.