UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Lecture 1 – Part 01 : Module Introduction

Module Introduction

- Module Outline
- Delivery
  - 2 hour **Programming lecture** weekly
  - 2 hour **Design Lecture** weekly
  - 2 hour **Programming tutorial** weekly
  - **Independent Study : 144 hours per Semester!!**
- Instructions
  - Attend (ALL) lectures and tutorials
  - Questions : please ask during the session. You can speak or use the chat
  - Try out Formative Assessments.
  - Rest of the guidelines, remain the same.

Module team – Lectures

- **Programming Lectures**
  - Pumudu Fernando (FT) - Module leader [IIT]
  - Torin Wirasinghe(FT/PT)
  - Lakna Gammedda (FT)
  - Dilshard Ahamed (FT)
  - Vishmi Embuladeniya (FT)
  - Ayoob Mohamed (FT)
  - Suresh Peiris (FT)

- **Design Lectures**
  - Torin Wirasinghe(PT)
  - Iresh Bandara (FT)
  - Roshan Gunathilake (FT)

Module team – Tutorials

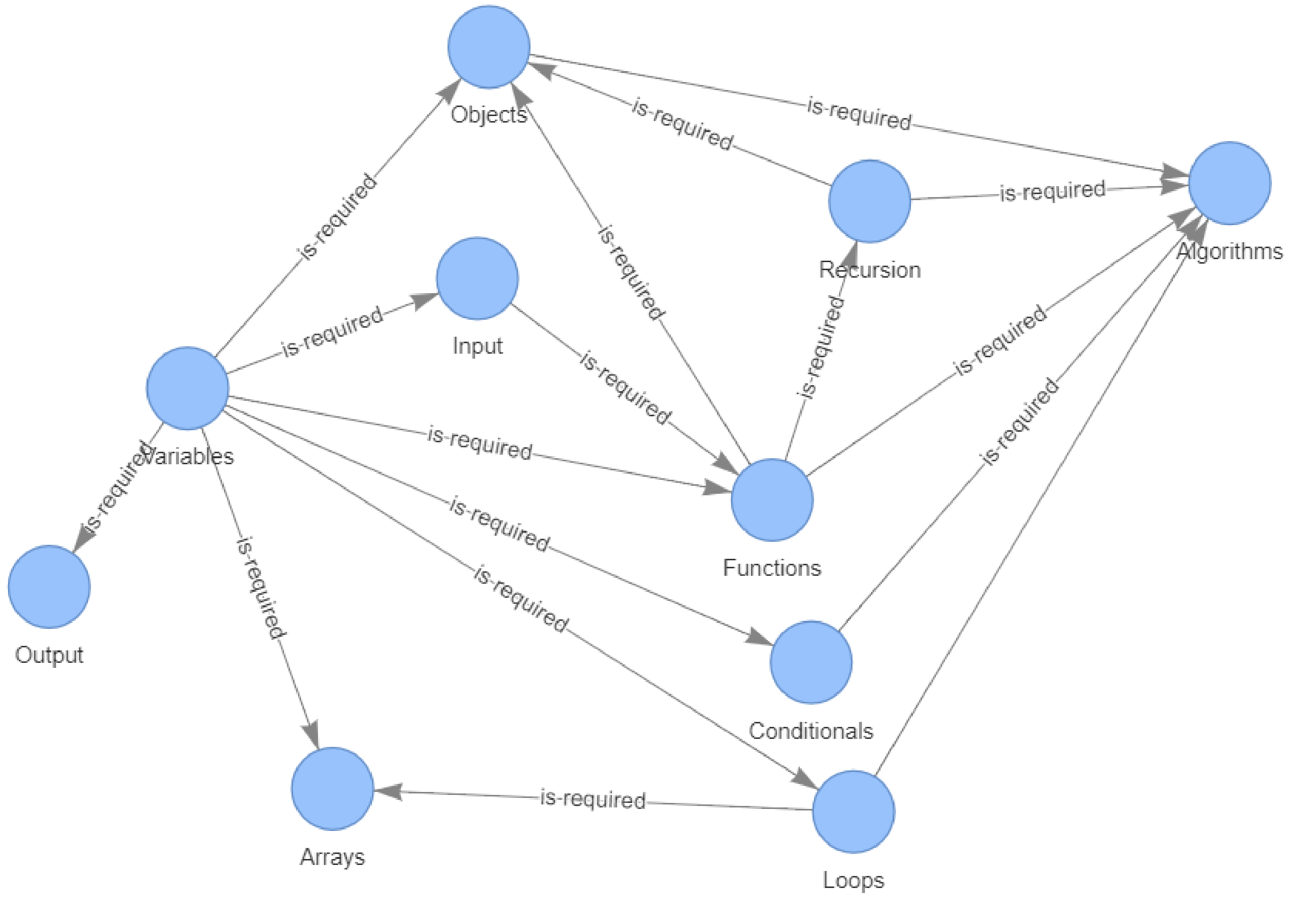
- **Tutorials**
  - Torin Wirasingha
  - Sulochana Rupasinghe
  - Dinusha Ruwankumara
  - Sapna Kumarapathirage
  - Ruwan Egodawatte
  - Lakna Gammedda
  - Kushan Bharethi
  - Kalhari Wewelage
  - Rashmi Perera
  - Raveen Sriskandharajah
  - Suresh Peiris
  - Abdul Baasith
  - Ruzaik Seyed
  - Adshayan Balachandran
  - Ammar Raneez
  - Dimithri Premachandra
  - Imesh Pathirana
  - Nazhim Kalam
  - Ruzaik Seyed
  - Salitha Perera

Assessment Structure

- **Assessment 1: Coursework (50%)**
  - Released: **Monday 12th February.**
  - Deadline: **Monday 18th March at 1pm.**
  - Coursework **VIVA**: **TBA**

- **Assessment 2** : **In Class Test (50%)**
  - Multiple choice test that will be conducted onsite (supervised) but over Blackboard.
  - Date will be informed later.
- **Click here for weekly schedule**

Module Content Connection



Required Software

Setting up the Java Development Environment

- Java Development Kit 17 or higher
    - Install for Windows | Mac | Linux

Code Editors

- Notepad++ [for week 1&2 Tutorials]
- Intellij IDEA [for week 3 tutorials onwards and Coursework]

Recommended References

- **Core Text and Essential Reading**
  - **Java for Everyone : Late Objects** by *Cay Horstmann*
    - https://www.oreilly.com/library/view/java-for-everyone/9781118063316/
  - **Book : Big Java** by *Cay Horstmann*
    - https://learning.oreilly.com/library/view/big-java-4th/9780470509487/

- **Additional Materials**
  - **Online Course** : **Java Essential Training for Students**
    - https://www.linkedin.com/learning/java-essential-training-for-students
  - **Online Course** : **Java Object-Oriented Programming**
    - https://www.linkedin.com/learning/java-object-oriented-programming-2

How to be Successful in this module

- During Lectures & Tutorials
  - Attend the sessions
  - Engage during the session
  - Ask and respond to questions
- During the tutorials
  - Read and analyze the question
  - Design and Code solution in Java
  - Test and improve the solution
  - Contact tutor if help is required

- During Independent Studies
  - Attempt weekly formative tests
  - Read recommended book chapters
  - Follow related online Video Tutorials
- During Assessments
  - Submit the coursework on time
  - Use formative tests to practice for ICT

The more you practice, the higher the chance to be successful in this module and following modules.

UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

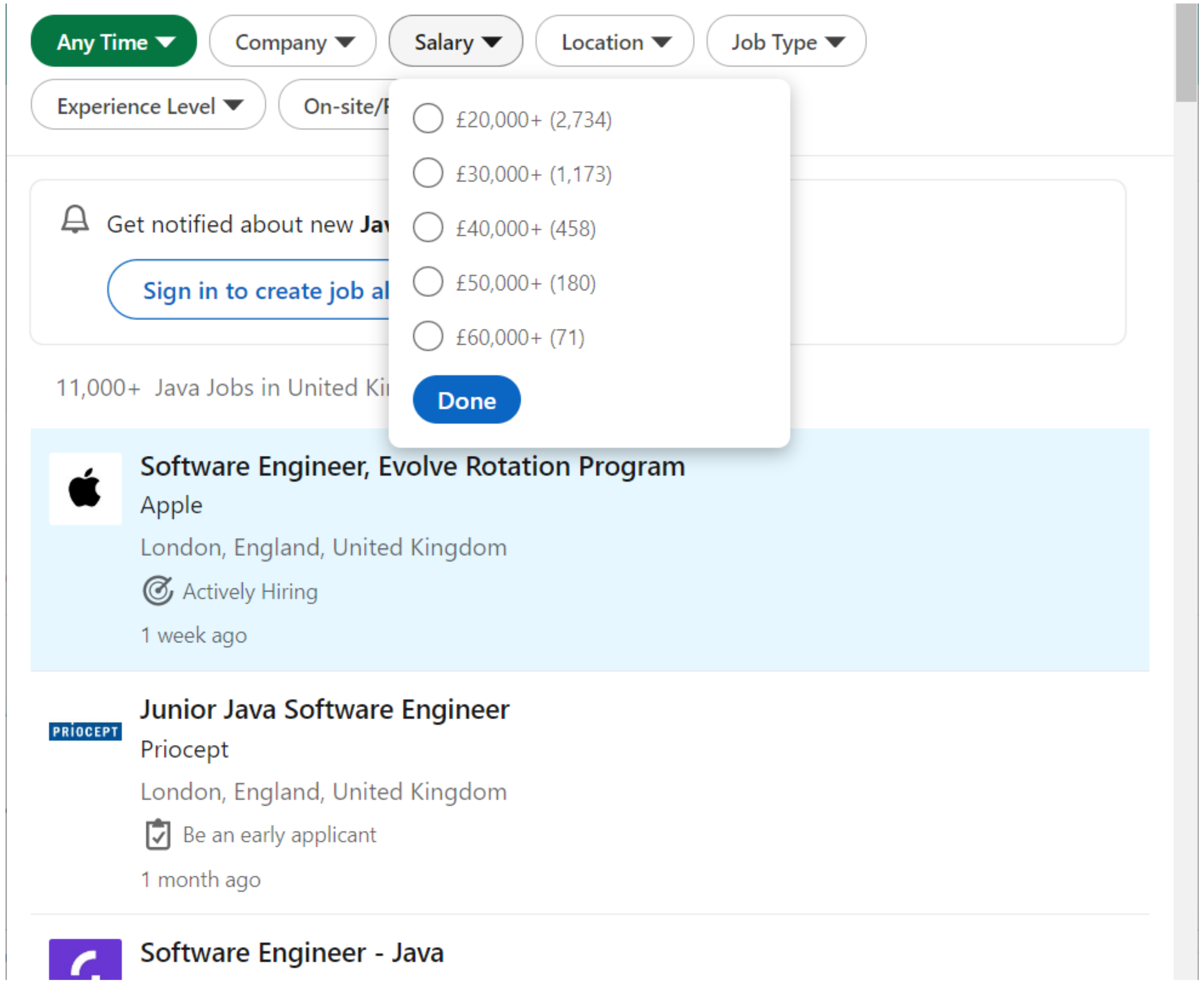
Lecture 1 – Part 02 : Introduction to Java

*Readings : **Java for Everyone** - Chapter 01*

JAVA

- Java is a programming language and computing platform.
- The most current is Java 21 (JRE 21 - Java Runtime Environment)
- Java is Object-Oriented--that means everything in the language behaves like an object.
- What exactly that means will be explained in the coming during the course.
- Java Documentation
  - https://docs.oracle.com/en/java/javase/19/docs/api/index.html

Why JAVA?


Any Time
Company
Salary
Location
Job Type
Experience Level
On-site/R

£20,000+ (2,734)
£30,000+ (1,173)
£40,000+ (458)
£50,000+ (180)
£60,000+ (71)
Done

Get notified about new Jav

Sign in to create job al

11,000+ Java Jobs in United Ki

Software Engineer, Evolve Rotation Program
Apple
London, England, United Kingdom
Actively Hiring
1 week ago

Junior Java Software Engineer
Priocept
London, England, United Kingdom
Be an early applicant
1 month ago

Software Engineer - Java

- Easier to learn than other languages and ideal to teach programming fundamentals.
- High demand: ~3,000 jobs in London (Search performed on 17/11/22)
- Safe and portable
- Wide range of development tools
- Great support
- Used for mobile applications, desktop applications, web applications, etc.
- Works on different platforms
- Open-open source and free

JDK, JRE and JVM

- **Java Development Kit (JDK)**

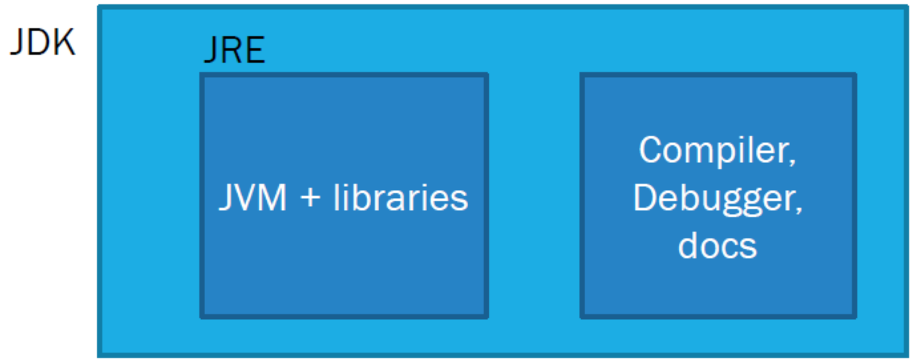
Is a software development environment for making Java applications. Contains tools required to write Java programs such as a compiler. Converts Java code to byte code.

- **Java Run Environment (JRE)**

Software that runs other software. It executes Java programs. Required to run Java programs. It also has class libraries (Math, util, etc.) and the JVM.

- **Java Virtual Machine (JVM)**

In Java, all code is run in a virtual machine, the JVM, so it is platform-independent. It is part of the JRE.



Java Architecture

Java's Architecture comes from four separate but intertwined technologies:

- Java Programming Language
- Java class file
- Java API
- Java Virtual Machine

Java Architecture contd…

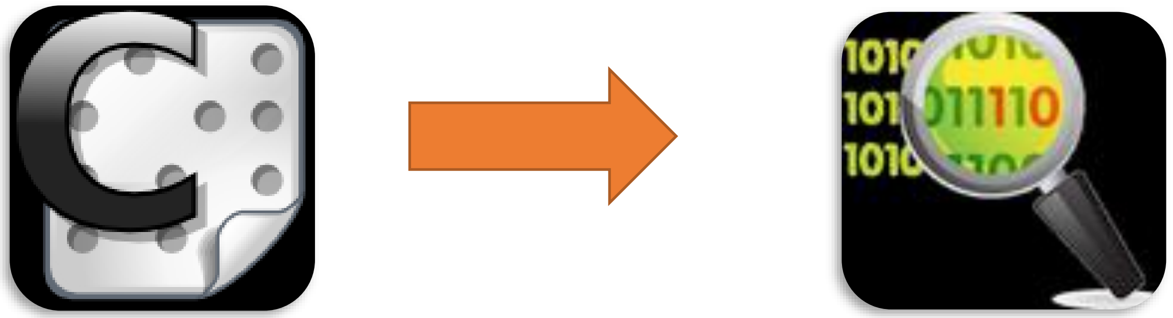
- Source programs are written in the Java Programming Language.
- Programs are compiled into Java class files.
- Classes run in the Java Virtual Machine.
- When a Java program runs, it is assisted by other classes in the Java the Application Programming Interface (API).

How to run a program in JAVA

- Edit source files
- Compile program
- Generate class files
- Run program

Typical Procedural Program

- In a typical C program, the source code is compiled into a native machine language module that consists of 1's and 0's.



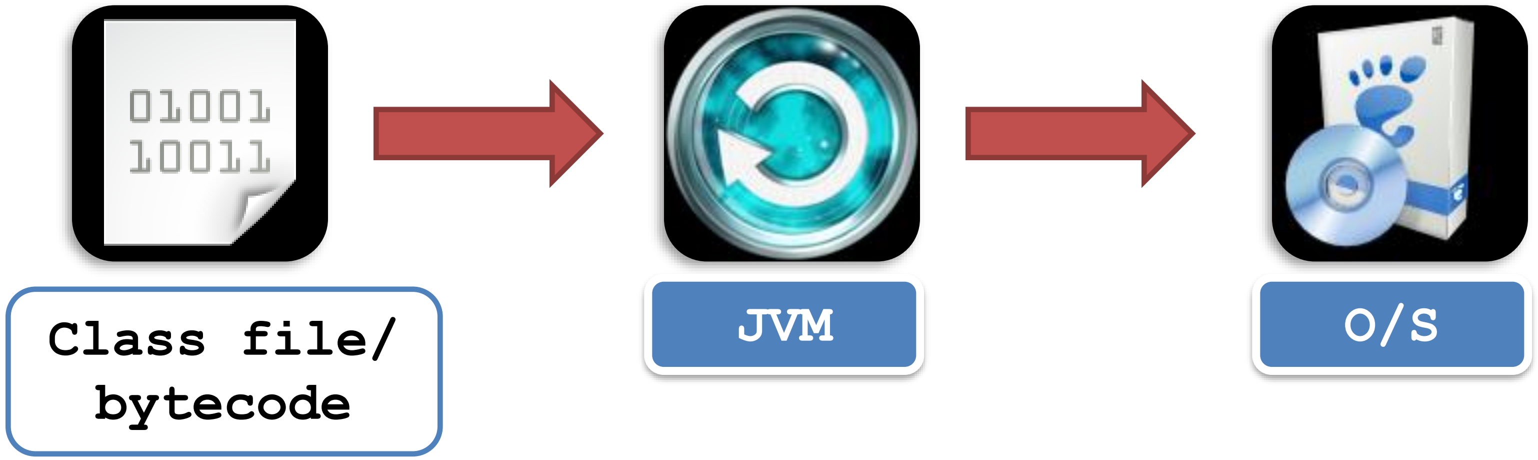
- The machine language is specifically tailored to one OS, be it Windows, Mac or UNIX.

Java Class file("Bytecode")

- In contrast to conventional programming languages, a Java program is not compiled into machine language.
- Instead, Java makes bytecode.
- Bytecode is the result of a Java "compile", a low-level code similar to machine language, but generic and not specific to any particular processor.
- Bytecode is been fed to the Java Virtual Machine (JVM) .

Java Virtual Machine (JVM)

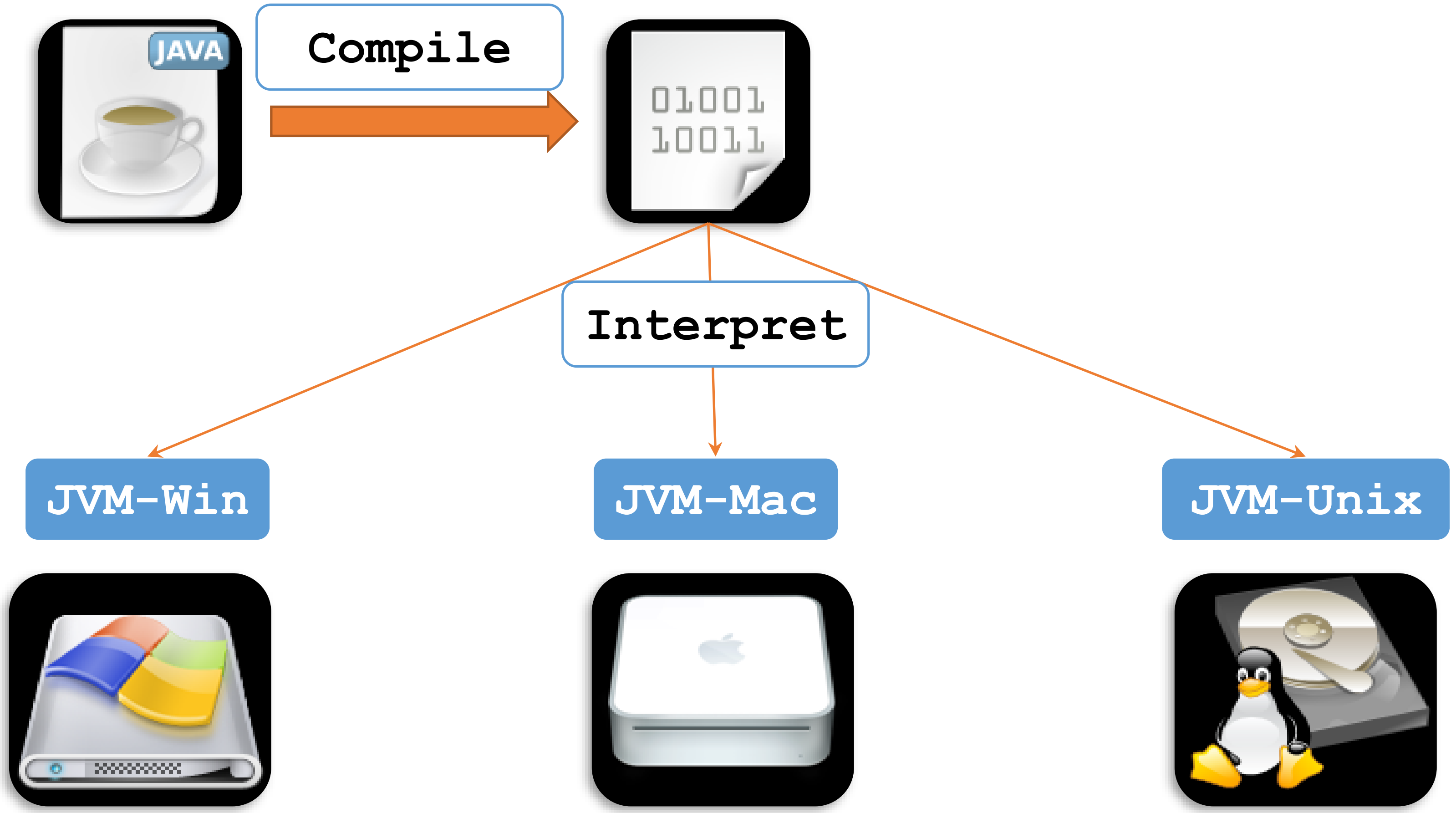
- The JVM is a software-only sub-computer within the OS that converts Java bytecode into machine language and executes.
- JVM is platform dependent so there are different JVM's for each OS.
- The bytecode talks to the JVM, and the JVM talks to the Operating System.



Java Class file(“Bytecode”)

- Java API (Application Programming Interface) is a set of classes and interfaces that comes with the JRE.
- It is a huge collection of library routines that performs basic programming tasks such as looping, displaying etc.

“Write Once Run Anywhere”

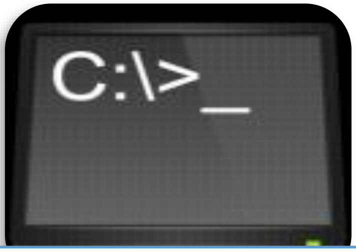


Types of Programs in Java

- A Java Application is a free-standing program that is capable of running directly in the Java Virtual Machine.
- A Java Applet is a mini-program that is much more limited in its abilities. An Applet can only run within the context of an HTML browser.

Types of Programs in Java

- Applications



Console Interfaces

Graphical User Interfaces

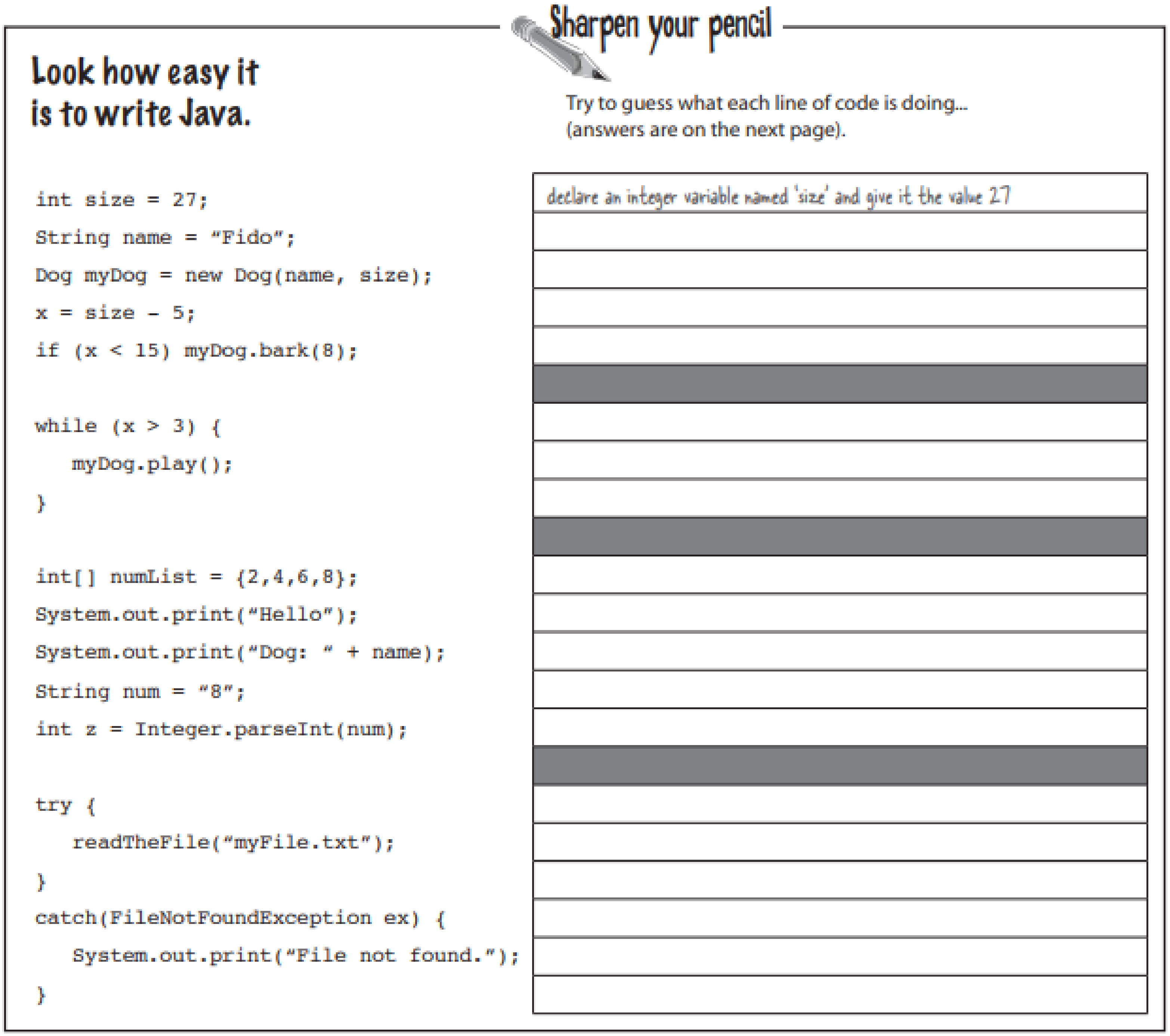
- Applets

What you need to run java

- Refer to this document for more instructions
- **Download and install Java Development Kit (select your OS)**

    https://www.oracle.com/java/technologies/downloads/

- Use a text Editor
    - Notepad or Notepad++
- Install an IDE (Optional)
    - **IntelliJ IDEA : https://www.jetbrains.com/idea/download/#section=linux (Recommended)**
    - Netbeans :https://netbeans.apache.org/download/index.html
    - Eclipse : https://www.eclipse.org/downloads/



Sharpen your pencil

Look how easy it is to write Java.

Try to guess what each line of code is doing... (answers are on the next page).

`int size = 27;`	declare an integer variable named 'size' and give it the value 27
`String name = "Fido";`	
`Dog myDog = new Dog(name, size);`	
`x = size - 5;`	
`if (x < 15) myDog.bark(8);`	
	▨
`while (x > 3) {`	
`   myDog.play();`	
`}`	
	▨
`int[] numList = {2,4,6,8};`	
`System.out.print("Hello");`	
`System.out.print("Dog: " + name);`	
`String num = "8";`	
`int z = Integer.parseInt(num);`	
	▨
`try {`	
`   readTheFile("myFile.txt");`	
`}`	
`catch(FileNotFoundException ex) {`	
`   System.out.print("File not found.");`	
`}`	

In class activity

Java vs. Python

- Java
  - Compiled and interpreted
  - Static-typed (variables types are known at compile time)
  - Relatively fast
  - Syntax is complex

- Python
  - Interpreted
  - Dynamic-typed (variable types are known at run time)
  - Relatively slow
  - Syntax is easy

Python (SD 1) vs. Java (SD 2)

Python

Code:

```
print("Hello")
```

==

Java

Code:

```
public class LectureMaterial {

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

Find the differences

Exercise from SD1 Tutorial Week 8

Code: **Python** Timestable.py

```
def timestable(number):

    for i in range(1,13):

        print(number*i, end=", ")


num = int(input('Enter a number: '))

timestable(num)
```

Output:

```
Enter a number: 7
7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84,
```

==

Code: **Java** Timestable.java

```
public class LectureMaterial {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = input.nextInt();
        timestable(number);
    }

    public static void timestable(int number){
        for (int i = 1; i < 13; i++){
            System.out.print(number*i + ", ");
        }
    }

}
```

Output:

```
Enter a number: 7
7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84,
```

The Java programming language

- Every Java program consists of one or more Classes (fundamental building blocks):
- `public class ClassName{}`
  - We will learn more about classes later in the module
  - The name of the file must be the same as the name of the class (classname.java)
  - Every Java application must contain a `main` method:

```
public class ClassName{
public static void main(String[] args) {
//Your code here
   }
}
```

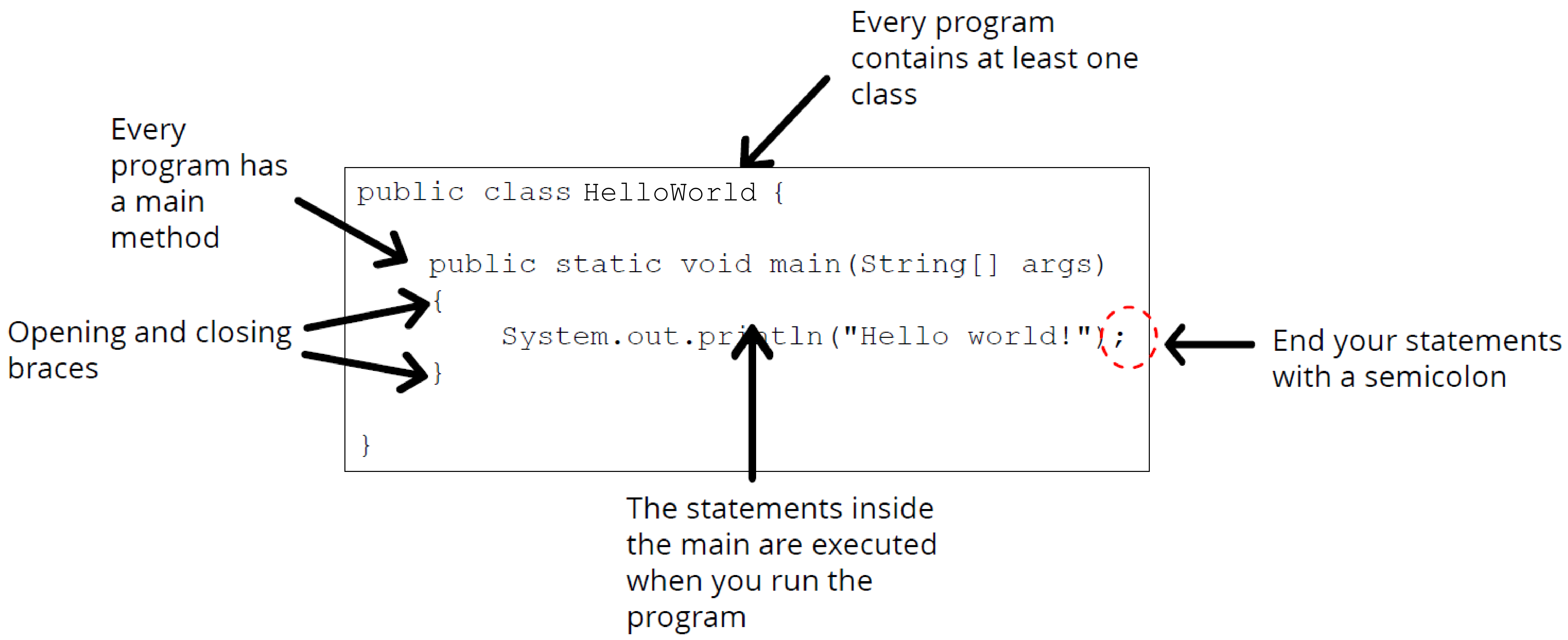
- There are public and private features (functions and variables)
- Case sensitive!
- Statements end with a semicolon ;
- Use opening and closing braces {}
- Needs to be compiled before running (the IDE will do this for you).

A Simple Java Application

- The double slashes denote a “C++” style comment. Everything on the line after the double slashes is ignored by the compiler.

```
// HelloWorld.java Our first Java Application
```

A Simple Java Application – Example 01

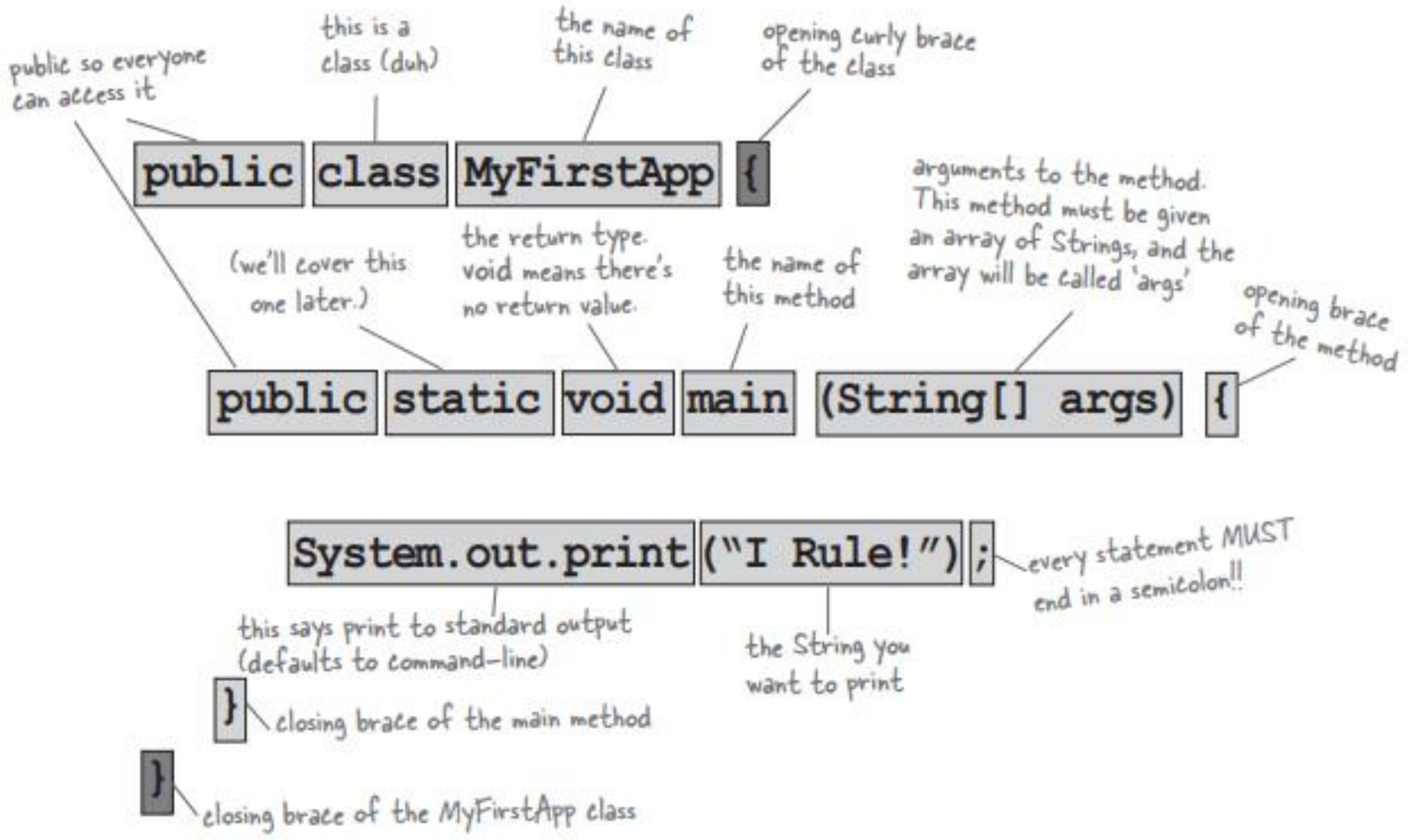


```
public class HelloWorld {

    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }

}
```

Every program contains at least one class
Every program has a main method
Opening and closing braces
End your statements with a semicolon
The statements inside the main are executed when you run the program

A Simple Java Application – Example 02

Examine the diagram below and understand by your self


public so everyone can access it
this is a class (duh)
the name of this class
opening curly brace of the class
public class MyFirstApp {
(we'll cover this one later.)
the return type. void means there's no return value.
the name of this method
arguments to the method. This method must be given an array of Strings, and the array will be called 'args'
opening brace of the method
public static void main (String[] args) {
System.out.print ("I Rule!") ;
every statement MUST end in a semicolon!!
this says print to standard output (defaults to command-line)
the String you want to print
}
closing brace of the main method
}
closing brace of the MyFirstApp class

How to Compile

```
C:\>javac HelloWorld.java

C:\>
```

- A successful compilation of your java program will return to a bare cursor, as you see here.

How to Execute

```
C:\>javac HelloWorld.java

C:\>java HelloWorld
Hello World!
```

- Note: the “.class” extension is omitted.

Packages and import

- The Java API is a library that contains numerous packages that you can use in your programs.
- A package is a directory storing classes and interfaces (files).
- For example, all interfaces related to input and output are stored in the java.io package.
- You can create a package using: package.
- To use a package, we have to import it using: import

Output

- We use outputs to: inform the user to take an action, to check that the program is doing what we expect to do, to display information, etc.
- To print an output, we use the class System: **`System.out`**
- To print: **`System.out.println()`**
- **`System.out.println(3+4)`**
- Will print **number**7.
- We need to be careful when printing numbers and characters together:
- **`System.out.print(“00” + 3 + 4 )//Concatonation`**
- Will print **text**‘0034’.
- Example:

```
public class ClassName {

    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }

}
```

Input

- To retrieve a value, you have to go through different classes.
- When you type a value in a program, to retrieve it, you can the **in** object of the **System** package:

  System.in

- After getting that value, you must first store it somewhere.
- One of the classes you can use is called **Scanner**.
- Before using the Scanner class, you must import the **java.util.Scanner** package into your program.

Input

We use inputs to input data in our program (e.g., keyboard).

To input using the keyboard, we use the class System: System.in

We also need to import a Scanner from the package java.util.

Example:

Import util

Create a Scanner

```
import java.util.*;

public class Hello {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Insert your name:");
        String name = input.next();
        String surname = input.next();
        System.out.println("Hello: " + name + “ “ + surname);
    }
}
```

Use Scanner

Use Scanner again

Compile Time Errors and Problem Solving

- When you forget a semicolon:

```
MyClass.java:7: error: ';' expected
        System.out.println("Hello world! ")
1 error
```

- When you forget a brace:

```
error: reached end of file while parsing
}
1 error
```

- When your class name does not match the file name:

```
MyClass.java:4: error: class MyClassName is
public, should be declared in a file named
MyClassName.java
public class MyClassName {
1 error
```

- If you use ' instead of "

```
MyClass.java:6: error: unclosed character
literal
        System.out.print('Hello world');
```

- You use System.out without capital letter (system.out):

```
MyClass.java:7: error: package system
does not exist
```

Find the error

Code:

```
public class MyClass {
    public static void main(String[] args) {
        System.out.print(Hello world!);
    }
}
```

Feedback: Formative test week 1 (Blackboard)

- Go to Blackboard and select the module
- In Learning Resources > Week 1 you will find a formative test to get feedback on the content of this lecture.

Questions?