Introduction to JAVA programming

Instructor: Nihshanka Debroy
What is a Programming Language?

- A vocabulary
- A set of syntactical (grammatical) rules
- Instructs the computer to perform specific tasks
- Can do almost anything in a programming language
- Particular languages encourage you to do things a certain way
- Examples: C, C++, Java, Perl
Layers of abstraction

The User:

Applications
Programming Languages
Operating Systems
Machine Architecture
Circuits
Physics

The Result:
Performance

- Performance: speed in completing some task
- Performance is everything to most computer and software manufacturers.

Story:
- If the transportation industry kept up with the computer industry, one would be able to now buy a Roll Royce that could drive across country in 5 minutes for $35.

Rebuttal:
- It would crash once a week, killing everyone on board.
What is JAVA?

- Some think its “Just Another Vague Acronym”
- Original name was “Oak”
- Popular programming language
- High-level language (closer to human language than machine language)

Image taken from Webopedia (http://www.webopedia.com/TERM/H/high_level_language.html)
JAVA

• Based on programming languages C and C++
• C++ newer and more complicated than C
• Start from C and add some of C++'s more useful features
Java programs

- Java programs are created as text files using a text editor (like emacs)
- **Save to disk with** `.java file extension` 
  ```
  HelloWorld.java
  ```
- **The file contains characters (stored as bytes)**
  - file can be printed, displayed on monitor, or edited
  - file cannot be directly executed (run) by the computer system
- **Java must first translate the program into bytecodes before it can be run**
Bytecodes

- **Java bytecode** → machine instruction for the Java processor
- **Java compiler javac** translates the source program into bytecodes
- **Bytecode file has same name as the source program with a .class file extension:** `HelloWorld.class`

![Diagram]

`HelloWorld.java` ➔ `javac` ➔ `HelloWorld.class`

source program  ➔  Java  ➔  Java bytecodes
Java Virtual Machine (JVM)

- Bytecode (class) file will contain exactly the same bytecodes no matter what computer system is used
- Bytecode file is executed by a Java bytecode interpreter
  - processor specific executable program
- Each type of computer system has its own Java interpreter that can run on that system
- Any computer system can execute Java bytecode programs if it has a Java interpreter
- Computers with Java interpreters are called Java Virtual Machines
  - a “computer” with a Java processor that can run Java bytecodes
First example: What do the parts mean?

```java
public class Example1 {

    /** @param args */

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

- Tells compiler you're creating a class called Example1 (so the java file should be Example1.java)
- Comments
- Function/method (set of statements grouped together), called main
- Beginning of main function
- Code statement: print a line of text
- End of main function
- End of class
Second example: Factorial

```java
int answer = 1;

answer = 2 * answer;
System.out.println("The answer is: " + answer);

answer = 3 * answer;
System.out.println("The answer is: " + answer);

answer = 4 * answer;
System.out.println("The answer is: " + answer);

answer = 5 * answer;
System.out.println("The answer is: " + answer);

answer = 6 * answer;
System.out.println("The answer is: " + answer);

answer = 7 * answer;
System.out.println("The answer is: " + answer);
```

- Integer variable Answer: starts as 1
- **Answer** = 2*Answer = 2
- **Answer** = 3*Answer=6
- **Answer**= 4*Answer=24
- **Answer**= 5*Answer=120
- **Answer**= 6*Answer=720
- **Answer**= 7*Answer=5040
- ; indicates end of line
public class Example3 {

    /**@param args*/

    public static void main(String[] args) {
        int answer = 1;
        int iterations = 9;
        System.out.println("Answer starts out as 1.");
        // initialization condition increment
        for(int i = 1; i <= iterations; i = i+1){
            answer = i * answer;
        }
        System.out.println("The value of " + iterations + "! is: " +answer);
    }
}
public class Example4 {

    public static void main(String[] args) {

        int iterations; // number of times we run through the loop
        int answer = 1; // holds the value we're calculating

        /* prompting the user for input

        System.out.print("Enter the number you want the factorial of, here-->");

        iterations = Keyboard.readInt();

        /* calculating the answer */

        for(int i = 1; i <= iterations; i++)
            answer = i * answer;

        // printing out the answer
        System.out.println("The answer is: "+answer);
    }
}