Lecture 1:
Introduction to Java

AITI Nigeria Summer 2012
University of Lagos.
Agenda

• First Lab ….. Class is Hands on remember ?
• Recap – Previously on AITI 😊
• What makes Java special?
• Advantages and disadvantages to using Java.
• Methodology for developing applications.
Recap - Teaching Style

• Emphasis on self-learning:
  – We will encourage you to discover your own answers
  – The most important skill you will ever learn

• Emphasis on participation:
  – Ask questions during lecture
  – Provide constructive criticism
  – Suggest course topics
  – Interrupt if we use jargon or idioms
Recap - Self-Learning

• Use MIT’s OpenCourseWare website to teach yourself Java
  • Website: http://ocw.mit.edu
  • ebooks
  • Why self-teach?
    – Move beyond the course curriculum
    – Develop a more advanced final project
    – We are here to help!
Recap - Student Evaluation

• There are no tests!
• Students will be evaluated on labs and projects:
  • Labs:
    – Design/Code
    – Output
    – Post-lab interview
  • Projects:
    – Idea
    – Milestone Presentations
    – Demo
Recap - Collaboration

- Students are encouraged to collaborate on labs and projects.

- However, copying code without understanding is not allowed.

- Zero tolerance
  - If found copying, .. Well, we are not sure if you belong in the class. Its always better to ask for clarification than to copy!!
**Starting Point - Compiler**

- A program that translates a programming language into machine code is called a *compiler*.

  ![Diagram](image)

  **High-Level Code**
  ```plaintext
  ... 
  a = b + c 
  ... 
  ```

  **Machine Code**
  ```plaintext
  ... 
  ld $r1, a 
  ld $r2, b 
  add $r3, $r1, $r2 
  st a, $r3 
  ... 
  ```

- Typically, we must have a compiler for each operating system/machine combination (*platform*).
Compiling Computer Programs

Because different platforms require different machine code, you must compile programs separately for each platform, then execute the machine code.
The Java Compiler is Different!

• The Java compiler produces an intermediate format called **bytecode**.

• Bytecode is not machine code for any real computer.

• Bytecode is machine code for a model computer.
  – This model computer is called the **Java Virtual Machine**.
Java Interpreter

• A Java *Interpreter* is required to execute the bytecode on a real computer.

• A Java Interpreter converts the bytecode into machine code.
  – As the program executes
  – *Simulate* the execution of the Java Virtual Machine on the real computer

• You can run bytecode on any computer that has a Java Interpreter (JRE) installed!
  – Only have to compile once
  – Can distribute the same bytecode to everyone
The Java Approach

Java Program → compiler → Java bytecode → Win Interpreter → Mac Interpreter → Unix Interpreter
Advantages of Using Java

• Once a Java program is compiled you can run the bytecode on any device with a Java Interpreter.
  – Because you do not have to recompile the program for each machine, Java is device independent.

• Java is safe. The Java language and compiler restrict certain operations to prevent errors.
  – Would you want an application to have total control of your phone?
    • Make calls, send SMS messages?

• Java standardizes many useful structures and operations such as lists, managing network connections, and providing graphical user interfaces
Disadvantages of Using Java

- Running bytecode through an interpreter is not as fast as running machine code
  - But this disadvantage is slowly disappearing

- Using device specific features (e.g., bluetooth) is difficult sometimes because Java is device-independent.

- In order to run a Java program on multiple devices, each must have a Java Interpreter
  - Ex: most Nokia phones come with Java Interpreter
Programming Methodology

1. Specify and analyze the problem
   - Remove ambiguity
   - Decide on inputs/outputs and algorithms

2. Design the program solution
   - Organize problem into smaller pieces
   - Identify existing code to reuse!

3. Implementation (programming)

4. Test and verify implementation

5. Maintain and update program
Writing Good Code

• A program that meets specification is not necessarily good.
• Will you be able to make changes to it?
  – Will you understand it after some time?
• Others might need to look at your code
  – Can they understand it?
• Write your program so that is easy to understand and extend!
  – Spend extra time thinking about these issues.
/* The HelloWorld class prints “Hello, World!” to the screen */
public class HelloWorld {
    public static void main(String[] args) {
        // Prints “Hello, World!”
        System.out.println("Hello, World!");
        // Exit the program
        System.exit(0);
    }
}

Comments

• **Comments** are used to describe what your code does as an aid for you or others reading your code. The Java compiler ignores them.

• Comments are made using `//`, which comments to the end of the line, or `/* */`, which comments everything inside of it (including multiple lines)

• Two example comments:
  - `/* The HelloWorld class prints “Hello, World!” to the screen */`
  - `// Prints “Hello, World!”`
Comments on Commenting

- You may collaborate on software projects with people around the world who you’ll never meet
- Should be able to figure out how code works by reading comments alone
- Anything that is not self-evident needs a comment
- 50% of your code might be comments
- Coding is easy, commenting is not