Lecture 1
JAVA (46-935)
Somesh Jha

Salient Features of JAVA

- Object Oriented
- Interpreted
- Portable
- Distributed
- Simple
- Robust
- Secure
- Multithreaded

Object Oriented

- All of you know what this means (Hopefully!)
- Think about data in the application and methods manipulating this data rather than procedures performing specialized tasks.
- OO programming changes the point of view from procedures to objects with attributes and methods.

Interpreted

- JAVA programs are first compiled into `class` files.
- `class` files are programs in a well specified language called the `bytecode`.
- Think about `bytecode` as an assembly language for an abstract machine.
- JAVA interpreter understands how to simulate a program written in bytecode. This is sometimes called the `JAVA Virtual Machine` or `JVM`.
- `JVM` is the abstract machine that `executes` programs written in `bytecode`.

Portability

- As long as there is a `JVM` on the platform you can run JAVA programs.
- It is very easy to write a portable program in JAVA.
- Non-portable programs do exist! (Especially programs with GUIs).
- JAVA programs can run on other devices such as SMARTCARDS.

Distributed

- JAVA interpreter dynamically loads classes as it needs them.
- Lot of high-level support for networking.
- Easy to write Client-Server programs.
- As a result of these capabilities one can run code from across the internet.
- For example, suppose a web browser needs to translate a file in Japanese to English.
- The browser can load a class (across the internet) to do the translation.

Simple

- No `goto` statement.

  `Substitute:` Labeled `break` and `continue` statements and exception handling.

- No pointers. This alone removes a lot of bugs.

  `Garbage Collection:` automatically reclaims memory that is not referred to.

- Syntax is also simpler than C++.

- Fewer kludges than C++ and hence a purer OO language.

Robust

- JAVA is a strongly typed language (more so than C++).
- Catch more errors at compile time.
- Lack of pointers removes a lot of bugs.
- Exception Handling is done in a much more systematic way.
- Exception handling deals with handling erroneous conditions (e.g. file not found).

Secure

- Security is important because JAVA is meant to be a distributed language.
- One can safely download a program from the internet and let it run on one's computer.
- Language restrictions (like lack of pointers) creates a much more secure language.
- Byte-code verification.
  - Before JVM runs a program it verifies certain properties about the program.
  - `(Example):` Execution stack doesn't underflow or overflow.
  - `(Example):` All bytecodes are legal.

Secure (Contd.)

- Sandbox.
  - Untrusted code is put in a sandbox where it has restrictions.
  - `(Example):` No access to the local file system.
  - Any core JAVA class that performs sensitive operations first asks permissions from the `SecurityManager` class.

- Digital Signature.
  Attach Digital Signature to the code and verify that the signature is of the entity you trust.

Multithreaded

- In a networked application (like a Web browser) several things can be going on at the same time.
- For example, a user might be reading the latest Financial news while some image (e.g. graph of bond yields) is loading up.
- JAVA provides clean support for multiple threads of execution.
- Benefit is improved performance. While a thread is waiting for input, another thread could be doing something useful.

Perfomance?

- JAVA is interpreted so it is slower than native code.
- In many networked applications speed is not a big issue.
- **Solutions**
  - Just-in-time compilers
    Convert bytecode programs to native code just before executing.
  - JAVA programs can call C and C++ programs.

What will we cover?

- Core JAVA.
- Client-Server programming in JAVA.
- Building Graphical User Interfaces (GUIs) in JAVA.
- `(Time permitting):` Other cool stuff.

Course Plan

- You will repeat some of the homeworks on option pricing in JAVA.
- We are developing a program for building an interest rate tree based on the BDT paper. This code will be used throughout the class.
- We will first present the basic code.
- We will turn the program into a client-server architecture.
- Then we will add a GUI on top of the program.
- Time permitting we will try to package the program so that it runs on a web browser.
- You will repeat the exact steps for the option-pricing code.

Course Evaluation

- Homeworks (4 or 5)
  60%
- Midterm (in the fourth lecture)
  20%
- Endterm (Last class)
  20%
- `Note:` Each exam will be 1.5 hours long.

Course Schedule

Number	NY (Wed)	London (Mon)
1	Oct 21	Oct 26
2	Oct 28	Nov 2
3	Nov 4	Nov 11 ($\star$)
4	Nov 9 ($\star$)	Nov 16
5	Nov 18	Nov 23
6	Nov 30	Dec 2
7	Dec 7	Dec 9

- London class (1:00pm-4:00pm)
- NY class (5:30pm-8:30pm)
- Midterm (Lecture 4)
- Endterm (Lecture 7)