Introduction to Programming

Lecture 1

Object-Oriented Programming

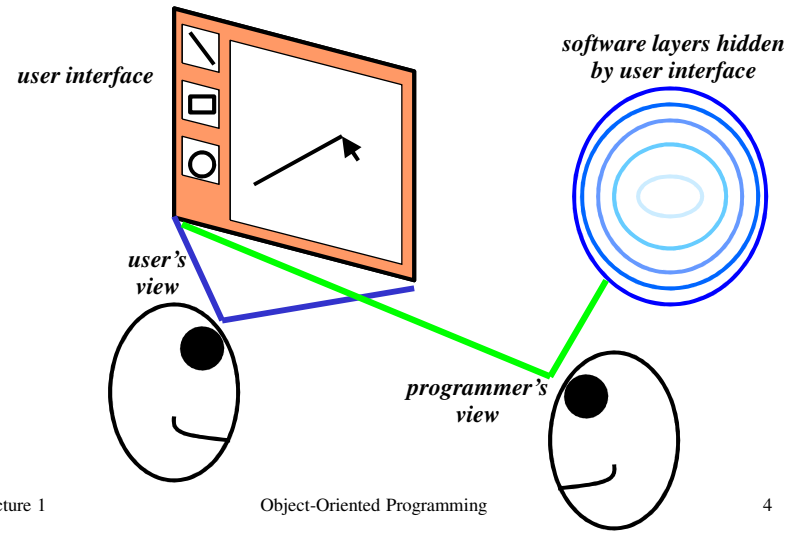
Agenda

- What is Program?
- Ideal Computing Way
- Where We Are in Computers?
- Interaction with the Computer
- Where are we going?
- Programming and its aspects
- Course Objective
- Object-Oriented Programming
- Interpreting vs. Compiling Programs
- Java Program Development Cycle
- Programming Tools
- Learn Programming
- Scare of Programming
- First Java Program
- Readings

What is a Program?

- Model of complex system
  - *model*: simplified representation of salient features of something, either tangible or abstract
  - *system*: collection of components that work closely together
- Sequences of instructions expressed in specific programming language:
  - *syntax*: grammatical rules for forming instructions
  - *semantics*: meaning/interpretation of instructions
- Instructions written (programmed/coded) by programmer
  - coded in a specific programming language
  - *programming languages* allow you to express yourself more precisely than *natural language*
  - as a result, programs cannot be ambiguous
  - all instructions together are called *source code*
- Executed by computer by carrying out individual instructions
- Real world example:
  - library catalog, word processor, video game, ATM

Two Views of a Program



The Ideal Way to Do Computing

- The ideal way to ask computer to do something is to order it in a natural language e.g.
  - I want to view this webpage
  - Calculate my annual tax
  - etc.
- However, today's computer's are not intelligent enough to understand our orders in natural language.

Where We Are in Computers?

- At the very basic level computers use the concept of an electrical pulse.
  - Low voltage is represented as 0
  - High voltage is represented as 1
- To instruct a computer we need ask the computer in the language of 0s and 1s commonly known as ***machine language***.
- For instance 73 in a number in natural language in the language of 0s and 1s, it becomes 1001001

Machine Language: Our First Interaction with the Computer

- Finding an average of two numbers in *machine language*.

```
10110011  00011001
01111010  11010001  10010100
10011111  00011001
01011100  11010001  10010000
10111011  11010001  10010110
```

- Not very intuitive way of working
- Not possible for humans to achieve a lot using machine language

One Step Beyond – Assembly Language

- One level above machine language is assembly language

```
MOV  0,  SUM
MOV  NUM,  AC
ADD  SUM,  AC
STO  SUM,  TOT
```

- More understandable but still very difficult for many of us.
- An *assembler* translates assembly language into *machine language*.

Another Step – High-level Languages

- High-level languages is another level above machine language.

  X = (Y + Z) / 2

- Much more understandable.
- A *compiler* translates high-level language into *assembly language.*

Where are we going?

- Te next step in computing is to use natural language over a high-level language.
- But we are many many years away from it.
- A lot of research needs to be carried out before we actually see this.
- Until then our task is to use high-level languages in its best possible ways

What is Programming?

- When we say “programming” we are actually referring to the science of transforming our intentions in a high-level programming language.

Many Aspects of Programming

- Programming is **controlling**
  - computer does exactly what you tell it to
- Programming is **teaching**
  - computer can only “learn” to do new things if you tell it how
- Programming is **problem solving**
  - always trying to make computer do something useful — i.e., finding an optimal travel route
- Programming is **creative**
  - must find a good solution out of many possibilities
- Programming is **modelling**
  - describe salient (relevant) properties and behaviors of a system of components (objects)
- Programming is **abstraction**
  - identify important features without getting lost in detail
- Programming is **concrete**
  - must provide detailed instructions to complete task

What are we doing in this course?

- Learn programming in a high-level programming language.
- Programming has many paradigms
  - Procedural
  - Object-Oriented
  - Functional
  - Logic
- We will study Object-Oriented Programming using ‘Java’, a popular high-level object-oriented programming language.

Object-Oriented Programming

- OOP: Now the dominant way to program, yet it is almost 40 years old! (Simula ’67 and Smalltalk ’72 were the first OOPLs)
  - Dr. Alan Kay received ACM's Turing Award, the "Nobel Prize of Computing,“ in 2003 for Smalltalk, the first complete dynamic OOPL
- It was slow to catch on, but since the mid-90’s everybody’s been doing it!
- OOP emphasizes objects, which often reflect real-life objects
  - have both properties and capabilities
  - i.e., they can perform tasks: “they know how to...”

OOP as Modeling

- In OOP, model program as collection of cooperating objects
  - program behavior is determined by *group* interactions
  - group interactions are determined by individual objects
- In OOP, objects are considered *anthropomorphic*
  - each is "smart" in its specialty
  - i.e., bed can make itself, door can open itself, menu can let selections be picked
  - but, each must be told when to perform actions by another object — so objects must cooperate to accomplish task
- Each object represents an *abstraction*
  - a "black box": hides details you do not care about
  - allows you as the programmer to control program's complexity — only think about salient features
- So, write programs by modeling problem as set of collaborating components
  - you determine what the building blocks are
  - then put them together so they cooperate properly
  - like building with smart Legos that you design!

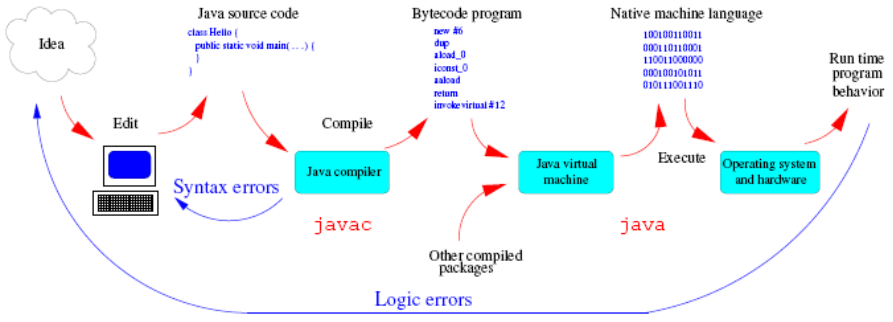
Interpreting vs. Compiling Programs

- An alternative to *compiling* your program is to *interpret* your program
  - each line of your program is translated into machine language and immediately executed
- Like translating between natural languages
  - **Compiler:** human translator translates book in its entirety and then translated book is printed and read
  - **Interpreter:** human interpreter translates each spoken statement in sequence as speaker is speaking

Running Java Programs

- Java uses both compilation and interpretation in a two-step process
- Compiles program into *bytecodes*
  - bytecode is close to machine language instructions, but not quite — it is a generic "machine language"
  - does not correspond to any particular machine
- *Virtual Machine* (VM) interprets bytecodes into native machine language and runs it
  - different VM exists for different computers, since bytecode does not correspond to a real machine
- *Same* Java bytecodes can be used on *different computers* without re-compiling source code
  - each VM interprets same bytecodes
  - allows you to run Java programs by getting just bytecodes from Web page
- This makes Java code run cross-platform
  - marketing says, "Write once, run anywhere!"
  - true for "pure Java," not for variants

Java Program Development Cycle



Programming Tools

- Key Tools for Programming
  - **Editors**: Allows user to enter the program. Notepad, Emacs, etc are all editors.
  - **Compilers**: Translates the program into target code (in machine language).
  - **Debuggers**: Allows a programmer to run the program to see the execution of the program and correct any errors.
  - **Profilers**: Used to evaluate program's performance.
  - **Integrated Development Environment (IDE):** Combines editor, compiler, debugger and profiler or a subset into one tool.
- Common Java IDEs are Eclipse, Netbeans, BlueJ, and DrJava.
- We will use DrJava IDE for the programming assignments in this course.

A Word of Caution

- IDEs, editors, debuggers, and other programming tools do not write program themselves. They merely provide some help in writing a program.
- Therefore,

THERE IS NO SHORTCUT TO PROGRAMMING SKILLS AND EXPERIENCE.

How to Learn Programming

- Everybody learns programming at their own pace.
- So do not be impressed by the person sitting next to you because he coded a given program in 20 minutes and you are taking more then an hour.
- Speed programming does not necessarily mean quality of the final output.

How to Learn Programming (cont'd)

- In a Nutshell
  1. Writing a good description of the problem.
  2. Breaking down the given problem into small pieces.
  3. Turning small pieces into pseudo-code.
  4. Deciding the integration mechanism of the pieces.
  5. Writing the program for each piece.
  6. Integrating all the pieces together.

Scare of Programming?

- Why most students are afraid of programming
  - Paradigm Change
    - Programming is totally different paradigm. You are working on something and you cannot even touch the final output you can only feel it. It is different then other subjects like Physics, Chemistry, Biology, etc.
  - Peer Pressure
    - Some people are naturally good in programming so others think that this is a natural ability and they cannot learn it.

Scare of Programming? (cont'd)

  - Lack of Understanding in Fundamental Concepts
    - Some people start programming without a clue of what is going on behind the scene in the computer. As a result they have a flawed understanding from day one of their programming experience
  - Time Factor: Programming takes a lot of time
    - Programming may take a lot of time at the start but once a person is comfortable with the concepts and has mastered the basic skills it is just like any other profession.

A Word of Advice

- Without good command on programming any qualification in Computer Science, Computer Engineering, Information Technology and Software Engineering is ***"worthless"***.
- There is an acute shortage of programmers in the global software market and with time this shortage is increasing.

First Java Program

```
class First
{

  public static void main(String arg[])
  {

    System.out.println("Engr. M Haroon Yousaf welcomes you in
     OOP");

  }

}
```

Readings

Book Name: Object-oriented Programming in JavaTM Textbook

Author: Richard L. Halterman

Content: Chapter 1

Book Name: Object Oriented Programming in Java – A Graphical Approach

Author: Kathryn E. Sanders & Andries van Dam

Content: Chapters 0

Acknowledgements

- While preparing this course I have greatly benefited from the material developed by the following people:
  - Andy Van Dam (Brown University)
  - Mark Sheldon (Wellesley College)
  - Robert Sedgewick and Kevin Wayne (Princeton University)
  - Mark Guzdial and Barbara Ericsson (Georgia Tech)
  - Richard Halterman (Southern Adventist University)