Introduction

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Software Overview

- **Two levels** → Operating System and Application
- **Operating system** → manages computer's resources; typically runs as soon as computer is turned on. Typical responsibilities:
  - Process management → Determines when, how programs will run on
  - Memory management
  - I/O, window system
  - Security
- **Applications** → programs users interact directly with; usually are explicitly run. Examples:
  - Word processors
  - Games
  - Music software
  - **Java Programs**
  - Etc
How Programs Are Executed

- Program “foo” initially stored in secondary storage
- Program copied into main memory
- CPU executes program instruction-by-instruction
Computers can only execute machine code

*Compilers are programs for translating programs (“source code”) into assembler / machine code*
Interpreters

- Another way to execute programs
  - Interpreters take source code as input
  - Interpreters execute source directly
  - Much slower than compiled programs
- *Debuggers* are based on interpreters
  - Debuggers support step-by-step execution of source code
  - Internal behavior of program can be closely inspected
- Common interpreter?
Object-Oriented Terminology

- Procedural-oriented languages
  - Programming centers around “actions”\textit{(verbs)}
- Object Oriented Languages
  - Centered around objects\textit{(nouns)}
- Objects
  - Principal entities that are manipulated by the program\textit{(nouns)}
- Class
  - A “blueprint”/recipe that defines the structure for one or more objects
- Method
  - Java term for a “function”, a “procedure”, or a “subroutine”
  - This is the code that does something \textit{(verbs)}
- Why we prefer the object-oriented approach?
  - One big reason: recycling
Tools for Writing Programs

• Good old days 😊
  • Text Editor → create source code files
  • Compiler → generate executables from source
  • Debugger → trace programs to find errors
  • Cycle between editing, compiling, running, debugging

• Today, IDE (Integrated Development Environment)
  • Editor, compiler, debugger, version control rolled in one

• IDE Examples
  • Eclipse → Free!! What we will use
  • Visual Studio
  • Netbeans
  • Others
Eclipse Fundamentals

• Eclipse IDE allow us to create, edit, compile, run, debug, programs
• Eclipse can be used for several languages; in this class Java
• Eclipse installation
• Eclipse tutorial
Eclipse Terminology

• Let’s run Eclipse
• **Workspace** → folder/directory where your files are stored
  • You can switch workspaces
• **Projects**
  • **Project** → collection of related files
  • Creating a program in Eclipse requires creating a project
  • Let’s create a project
Eclipse Terminology

• Perspective
  • Framework for viewing/manipulation programs
  • Three important perspectives
    • Java → creating, running programs
    • Debug → tracing, removing errors in programs
    • CVS repository → accessing/managing project files
  • Let’s change perspectives
  • Resetting perspective
• Let’s create a program