What is Programming?

Aspects of Programming, Computer Languages, Objects and Object-Oriented Programming

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 the computer do something useful, e.g. 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
- Programming is a craft
  - A bit like architecture, engineering — disciplined and creative craft for building artifacts

Many Aspects of Programming

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

What’s a Program? (1/3)

- Model of complex system
  - o model: simplified representation of salient features of something, either tangible or abstract
  - o system: collection of collaborating components

What’s a Program? (2/3)

- Sequences of instructions expressed in specific programming language
  - o syntax: grammatical rules for forming instructions
  - o semantics: meaning/interpretation of instruction
What’s a Program? (3/3)

- Instructions written (programmed/coded) by programmer
  - coded in a specific programming language
  - programming languages allow you to express yourself more precisely than natural (human) language
  - as a result, programs cannot be ambiguous

- Real world examples
  - Banner, word processor, video game, ATM, smartphone, browser

- Executed by computer by carrying out individual instructions

Java Programs

- CS15 and CS16 use Java
  - Java was developed by Sun Microsystems (now part of Oracle)
  - it is meant to run on many “platforms” without change, from desktop to cell phones
  - platform independence works quite well
  - but Java isn’t sufficient by itself: many layers of software in a modern computer

The Computer Onion

- Layers of Software
  - cover hardware like an onion covers its core
  - make it easier to use computers
  - organized into libraries and programs

In CS15, we only deal with the outermost layers

Two Views of a Program

Software layers hidden by user interface

User's view

User interface

Programmer's view
Programming Languages (1/3)

- Machine language
  - machine is short for computing machine (i.e., computer)
  - computer’s native language
  - sequence of zeroes and ones (binary)
  - different computers understand different sequences
  - hard for humans to understand:
    - 01010001...

Programming Languages (2/3)

- Assembly language
  - mnemonics for machine language
  - low level: each instruction is minimal
  - still hard for humans to understand:
    - ADD.L d0, d2
  - you’ll learn assembly language in CS33

Programming Languages (3/3)

- High-level languages
  - FORTRAN, C, C++, Java, C#, Python, JavaScript, Scheme, Racket, Pyret, ML, etc.
  - high level: each instruction is composed of many low-level instructions
  - closer to English and high school algebra
  - easier to read and understand
  - hypotenuse = Math.sqrt(leg1 * leg1 + leg2 * leg2);

Running Compiled Programs (1/2)

- In CS15, code in a high-level language, Java
- But each type of computer only “understands” its own machine language (zeroes and ones)
- Thus must translate from Java to machine language
  - a team of experts programmed a translator, called a “compiler,” which translates the entirety of a Java program to an executable file in the computer’s native machine language.
**Running Compiled Programs (2/2)**

- Two-step process to translate from Java to machine language:
  - **compilation**: your program → executable
  - **execution**: run executable
  - Machine executes your program by “running” each machine language instruction in the executable file
  - Not quite this simple “underneath the covers” – “Java bytecode” is intermediate language, a kind of abstract machine code

---

**Object-Oriented Programming (1/2)**

- **OOP**: Now the dominant way to program, yet it is over 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
  - OOP was slow to catch on, but since mid-90’s everybody’s been using it! But it isn’t the only useful programming paradigm...

---

**Object-Oriented Programming (2/2)**

- OOP emphasizes objects, which often reflect real-life objects
  - Have both properties and capabilities
  - i.e., they can perform tasks: “they know how to…"
- Look around you… name that object!

---

**OOP as Modeling (1/3)**

- 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
  - E.g., 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
OOP as Modeling (2/3)

- Each object represents an abstraction
  - a "black box": hides details you do not care about
  - allows you as the programmer to control programs' complexity - only think about salient features

OOP as Modeling (3/3)

- So, write programs by modeling problem as set of collaborating components
  - you determine what the building blocks are
  - put them together so they cooperate properly
  - like building with smart Legos, some of which are pre-defined, some of which you design!

Example: Tetris (1/3)

- What are the game’s objects?
- What do those objects know how to do?
- What properties do they have?

Example: Tetris (2/3)

- What are the game’s objects?
  - piece, board
- Capabilities: What do those objects know how to do?
  - piece
    - be created
    - fall
    - rotate
    - stop at collision
  - board
    - be created
    - remove rows
    - check for end of game
Example: Tetris (3/3)

- Properties: What attributes and components do they have?
  - piece
    - orientation
    - position
    - shape
    - color
  - board
    - size
    - rows

Software Development: A 5-Step Process (1/3)

1. Analysis
   a. English description of what the system models to meet user requirement/specification
2. Designing the system
   a. "Divide et impera" - divide and conquer: system is composed of smaller subsystems which in turn may be composed of even smaller subsystems (diagrams often helpful)
3. Implementing the design (in Java for CS15)
   a. if design is good, most of the hard work should be done
4. Testing and Debugging
   a. testing: submitting input data or sample user interactions and seeing if program reacts properly
   b. debugging: process of removing program bugs (errors)
5. Maintenance
   a. in a successful piece of software, keeping a program working and current is often said to be 80% of the effort

Software Development: A 5-Step Process (2/3)


   ● Good program
     - solves original problem
     - well structured, extensible, maintainable, efficient,… and met deadline and budget constraints…
     - Other developmental processes exist (e.g., extreme programming)
### Announcements (1/2)

- If you are even considering taking the course, we need you to register on Banner before next Monday (9/14) at 5:00pm – our first lab starts the next day!
- Introductory lab sessions will begin next week in the Sunlab (CIT 143). Meeting times are:
  - Tuesday: 5:00pm-6:30pm, 6:30pm-8:00pm, 8:00pm-9:30pm
  - Wednesday: 6:00pm-7:30pm, 7:30pm-9:00pm, 9:00pm-10:30pm
  - Thursday: 10:30am-12pm, 4:00pm-5:30pm, 5:30pm-7:00pm, 7:00pm-8:30pm
- Later today, we will email you a form where you will be able to register for a session, so check your email!

### Announcements (2/2)

- RISD and other non-Brown students please come speak to an HTA or Andy after class
- Check the course website at [http://www.cs.brown.edu/courses/cs015](http://www.cs.brown.edu/courses/cs015) and your email regularly.
- If you are undecided about which CS intro course to take, this document is a good reference: [https://cs.brown.edu/degrees/undergrad/whatcourse/](https://cs.brown.edu/degrees/undergrad/whatcourse/)