Introduction To Java Programming

Chapter 1
Computer Science

Main activities

- Misconception: Computer Science = Programming
- **Computer scientists solve problems**
  - Solutions expressed as an algorithm
    - “A step-by-step description of how to accomplish a task”
    - A recipe
  - Solutions expressed as a program
    - “A list of instructions to be carried out by a computer”
    - Source code is written in a human-readable language
    - Machine code is a binary computer-readable language
Computers

The stored-program architecture

- CPU
  - Control Unit
  - Registers
  - ALU

- System bus
  - Main Memory

- Disk controller

- NIC

- Video controller

---

CS 1400  Chapter 1  Slide 3 of 34
Language Levels

Where do they all go?

- High-level
  - Close to problem
  - System independent

- Low-level
  - Close to system
  - Doesn’t reflect problem

Java, C#

FORTRAN, COBOL, C++

C/C++

Assembler

Machine
Programs

Control

- Computers are generic tools
  - They become a specific tool when running a program
  - Programs drive the CPU, memory, etc.
  - CS 1400 focuses on programs (written in the Java programming language)

- The instruction execution cycle
  - Programs are converted into machine instructions
  - Fetch the next instruction from main memory
  - Decode the instruction into primitive electronic operations
  - Execute the instruction – route the operation to the appropriate hardware and carry out the operation
Programming Languages

Language elements

- **Symbols**
  - Key words – have special meaning (e.g., class, for, while, p. 17)
  - Operators – perform an action (e.g., +, *, /, .., new, instanceof)
  - Blocks (code contained by ‘{’ and ‘}’, p. 11)

- **Programmer named elements (identifiers, p. 16)**
  - Classes (p. 10)
  - Methods (AKA functions, p. 11)
  - Variables (instance and local, p. 63)
  - Library classes/methods (AKA programming API)

- **Syntax**
  - Rules for putting the program together
  - Expressions
  - Statements
Expressions and Statements

The building blocks

- Expressions produce a value (p. 55)
  - Constant
  - Variable
  - Method call (returning a value – not void)
  - Grouping of the above combined with operators
  - Form r-values (allowed on the RHS =): \( y + z \) or \( \text{sqrt}(y) \) or \( n > 100 \)

- Statements form a complete unit of work (p. 12)
  - Are *terminated* with a semicolon (;)
  - May span lines or may be more than one on a line
  - Method call (not returning a value or value ignored)
  - Assignment statements: \( x = y + z \); or \( p = \text{Math.pow}(x, y) \);
  - Compound (or block) statements are formed of simple statements between \{ \) and \} (block is not terminated with a semicolon)
Compiling Programs

Multi-file programs

editor -> source code

compiler

object files

linker

libraries -> executable

runtime

.cpp

.obj
How Does Java Work?

Producing portable code

- Source code is written in the Java Programming Language
- Source is compiled
  - The compiler converts Java source code into “byte code”
  - `javac prog.java` (creates `prog.class`)
- Byte code is machine code for a non-existent or virtual machine (Java Virtual Machine – JVM)

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```
Compiling and Running Java

Multi-file programs

- Source code
- Compiler
- Byte code

Java class libraries (API)

JVM

- Other packages

- Class loader

- Editor

- .java

- javac

- .class

- Java
The Two Lives of Java

Running Java Programs

- Virtual code runs on a Java virtual machine (JVM)
  - An application is a local program executed on a local JVM
  - An applet is a program downloaded from the net and executed on a JVM that is a part of the web browser

```
java prog (reads prog.class)
<br>
<applet code="prog.class"></applet>
```
Hello.java : An Example

The quintessential first program

public class Hello // class name & file name match
{
    public static void main (String args[ ])
    {
        System.out.println("Hello World");
    }
}

javac Hello.java // Note that Java is case sensitive
java Hello
Comments

Source code documentation

- Three styles of comments
  - /* comment */
  - // comment to end-of-line
  - /** embedded “javadoc” documentation comment */

- Different styles may be nested (if you’re careful)
  - Temporarily “comment out” code containing comments
  - Leave debug or development statements in source code but exclude them from compiled production code

- Only use javadoc comments for embedding documentation
The Java System

Summary

- Java System Files
  - Source code files (names end with .java)
    - Contain classes
    - At most one public class may be defined in a file
    - The file name is the same as the public class name
  - Bytecode files (names end with .class)
    - Contain virtual machine code
    - A .class file is created for each class defined in the .java file

- Java organizes related classes into packages
  - Each package occupies a separate subdirectory
  - Package names and subdirectory names are the same
  - Programs access the contents of a package by importing
    - import packagename.classname;
    - import packagename.*;
Debugging

Correcting errors

- If debugging is the process of taking bugs out of a program, then programming is the process of putting bugs in to a program.

- “As soon as we started programming, we found out to our surprise that it wasn’t as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.”
  – Maurice Wilkes, p. 21

- “Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”
  – Brian Kernighan (Kernighan’s Law – at least one version)
Bug Categories

Kinds of errors occurring in a program

- **Syntax error**
  - Detected at compile time (aka compile-time error)
  - Violates the language’s (syntax) rules for specifying a program
  - Compiler gives a line number – look there and above

- **Runtime error**
  - Detected when the program executes or runs
  - Program crashes and burns
  - JVM will print a stack trace – find the line number of the first reference to your code; use print and assert statements

- **Logical error**
  - Detected ??????
  - Program produces incorrect results or behaviors
  - Compare intermediate results with a hand-worked solution
Functional Model

Oldest model (aka functional or procedural decomposition)

- Focuses on how (i.e., the algorithms) to solve a problem
- Decomposes problem into functions and function interfaces
  - See diagram, p. 27
- Changes have wide spread effects – global data makes the program fragile
- Functions have varying degrees of coupling through global data
  - Developed as a unit
  - Debug as a unit
  - Validate as a unit
Functions
(aka methods, subroutines, subprograms, procedures)

```c
int a = 100;
int b = 25;
```

- When used or called, a function can be treated as a black box
  - User cares only about what the function does, not about how (performance and errors aside)

- Pass-by-value input (parameters) may be
  - Constants
  - Variables
  - Expressions
Functions

(methods, subroutines, subprograms, procedures)

Function call looks like a sequential statement (caller treats it like a black box).

Function definition must be concerned with call sequence. Execution resumes with the statement following the call.
Data Driven Models

‘70s through mid ‘80s

- Data flow
  - Maps data input to data output
  - Design data structures first
  - Design processes / functions last

- Data hiding
  - Packages data and the functions that work on the data together in a module (a file in C)
  - Data is still in global scope but access is allowed only through the module functions

- Abstract Data Type (ADT)
  - Programmer created data type
  - A class in Java
Object Model

State of the art

- Characteristics of functional & data models
- A tool for managing complexity
- Change resilient
  - Change is localized
  - Intra-object functions may be coupled
  - Extra-object functions are decoupled
- Natural organization for data and functions
  - Objects *encapsulate* data and functions together
  - Supports ADTs: multiple objects of a type may be created (class is a type specifier or ADT)
  - Supports data hiding: data access is controlled through key words

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Attributes</th>
<th>Functions Behaviors Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Notation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object-Oriented Model

The big picture

“Object-oriented modeling and design is a new way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity.”

James Rumbaugh, *Object-Oriented Modeling and Design*

- **Data Structure** (more commonly called an attribute)
  - variable
  - data member
  - instance field
  - instance variable
  - data
  - data field
  - state

- **Behavior**
  - method
  - member function
  - function
  - operation
  - sending a message is equivalent to calling a method
  - service
Classes

Describe objects

- Template, blueprint, or abstraction of a real-world “thing”
- A type specifier (i.e., an Abstract Data Type or ADT)
- Implements encapsulation and data hiding (modifiers provide explicit access control)
- Defines relationships with other classes
  - Inheritance (AKA generalization) — “IS A”
  - Association (connects peers, bidirectional — “HAS A”)
  - Composition (whole-part, tight — “HAS A” or “IS A PART OF”)
  - Aggregation (whole-part, loose — “HAS A” or “IS A PART OF”)
  - Dependency (AKA using or delegation – short-lived connection)
Class Example

The fundamental program unit

```java
public class Circle {
    private int radius; // instance variables
    private int x;
    private int y;

    public Circle(int r, int ax, int ay) { // constructor
        /* body */
    }

    public void draw() { // method
        /* body */
    }
}
```
Objects

Created from classes

- Objects are instances of classes
  - A class is a type and an object is a variable of that type
  - `new Circle(100, 20, 10);`
    - Allocates memory
    - Calls the Circle constructor method and passes it three parameters
    - Returns the address of the new object
    - 3 cases where objects are not created with new: string literals, statically allocated arrays, and factory methods

- Objects are accessed through reference variables
  - `Circle myCircle;`
  - Aliases are variables that reference the same object
    - Created by assignment
    - Created by method (function) calls
Object Examples

Objects and reference variables

Circle myCircle;

myCircle = new Circle(100, 20, 10);

Circle myCircle = new Circle(100, 20, 10);

Circle temp = myCircle;
spin(myCircle);

void spin(Circle aCircle) { }
Java Language Features

White paper @ http://java.sun.com/docs/white/langenv/

- Multiple levels of security
  - Memory protection (no explicit pointers and no pointer arithmetic)
  - Dynamic (runtime) code verification
  - Code origination monitoring (local code trusted more than imported)
  - Signed applets

- Intrinsic
  - Multithreading
  - Networking
  - Graphics and graphical user interfaces (GUIs)

- Java is robust
  - Exception handling
  - Catches many errors at compile time
The Story of Java

Chapter 1

- James Gosling & project green
- People expect computers to crash but not their appliances!
- Consumer electronics
- Powerful but error prone

C++
The Story of Java

Chapter 2

- Gosling created a new language called “Oak” after the large tree he could see through his office window.

- Oak was a safe language, a language that ushered in kinder and gentler programs. Programmers, users, and small animals were now safe.
The Story of Java

Chapter 3

- However, the name Oak was already taken, so they began calling their new language Java™ (for all of the beverage they were drinking at the famous coffee shop across the street from their office)

- Java™ has delivered on most of its promises
  - It is safe and secure
  - It is portable and runs on most computers and on most operating systems
  - It is embedded in consumer electronics, from cell phones to organizers
Java’s Roots

Influential ancestors

- Smalltalk
  - Pure OO (all data & functions in a class)
  - Classes derived from class Object
  - Interpreted
  - Garbage collection

- C++
  - Similar intrinsic data types
  - Identical control structures
  - Punctuation and syntax

- XEROX GUI
Java Is Object-Oriented

What are objects?

- Objects have attributes and behaviors
  - Attributes are data or variables
  - Behaviors are functions, procedures, methods, or subroutines
- Objects *encapsulate* data together with methods (i.e., functions)
  - The object is “responsible” for the data and protects it by locking it away
  - The object only exposes its methods, data are only accessed via these methods
- Java is a pure object-oriented language
  - Everything is defined inside of a class
  - Classes describe objects
Programming With Objects

How are objects used to program?

- Each object is a building block that has responsibility for one, small part of a program

- Programs are assembled from many, cooperating objects, which “snap” together