

**Class 1: Introduction**

**Introduction to Computation  
and Problem Solving**

**Prof. Steven R. Lerman  
and  
Dr. V. Judson Harward**

**Handouts for Today**

- **Course syllabus**
- **Academic Honesty Guidelines**
- **Laptop request form**
- **How to install Java and Eclipse on your laptop**
- **How to turn in your homework**
- **Lecture 1 notes**
- **Problem set 0**

## **1.00/1.001 course information**

- **Course staff:**
  - 2 instructors, 4 TAs, 2 lab TA, graders
- **Course Web page:**
  - All course information on Web
  - Lectures, labs, tutorials, problem sets
  - Hardcopy handouts at lecture, lab, tutorial
    - Pick them up as you come in
- **Grad students: register for 1.001, not 1.00**

3

## **Course goals**

- **Core concepts of software development**
  - Software design and requirements
  - Development and debugging/testing
  - Teamwork in software implementation
- **Programming in interactive, object oriented environment:**
  - Java; Microsoft C# is very similar
  - Very brief intro to C++ and C# at end
- **Use of computation for scientific, engineering, management problems**
  - Homeworks cover variety of problems
- **Software patterns**

4

## **Course goals, p.2**

- **Graphical user interfaces**
  - Java Swing, event models
- **Algorithms**
  - Sorting and searching
  - Concepts, programming, libraries
- **Data structures**
  - Stacks, queues, trees, lists, ...
  - Concepts, programming, libraries
- **Use of libraries**
  - Prewritten modules for common tasks

5

## **Laptops, labs and tutorials**

- **Limited number of laptop loaners for those who don't have them; softw re help for those with own laptops**
- **Signup for tutorial sections on line**
- **Get laptop today or Friday**
- **The first session in class using laptop is Tuesday, Sept 13 – Attendance at class lab sessions part of grade**
- **Tutorials start next week: Mon and Tue**
  - Attendance part of grade
  - Come with laptop after first tutorial

6

## **Laptops, labs, tutorials (p.2)**

- **Labs use laptops, active learning exercises**
  - **Mini-lectures with lab exercises: programming, simulations, short exercises.**
  - **TAs, instructors assist during lab**
- **Tutorials will also use active learning methods**
  - **About 10 students per tutorial;**
  - **Short questions, review lecture topics, design exercises**
- **Loaner Laptops**
  - **Loaner laptops have wireless. Use them all over campus**
  - **You may use them for other classes this term**
  - **Return them at or before final exam**
- **Homework**
  - **Turn in**

7

## **Wireless Laptop Initiative**

- **Course 1.00/1.001 was one of 4 wireless laptop pilot projects**
- **Why laptops? Some reasons:**
  - **Easy, convenient access to computing**
  - **Assess value of collaborative learning**
  - **Examine supportability of this technology**
- **“use your own” + loaner laptops**
- **Wireless cards available on loan**

8

## Writing Java programs

- Laptop computers (Microsoft Windows XP)
  - Eclipse integrated development environment (IDE).
  - You should load Eclipse on your own laptop or desktop computer:
    - Windows2000 or XP, 256MB RAM or more strongly recommended
  - Lab on Tuesday and tutorials next week will teach you how to use the Eclipse IDE

9

## Course Requirements

- 10 problem sets (48% of grade). Usually due on Fridays, but see calendar on web site
- 2 in-class quizzes (20% total) in regular class time. No evening quizzes.
- Final exam during finals period (22%)
  - Quizzes, exam are open book, open notes
- Tutorials and labs. Mandatory attendance. (10%)

10

## Course Resources

- **Lab TA hours: Wed and Thu evening, about 8 hrs each**
- Instructor office hours T/R (Harward) and Thurs 3:30 Thurs (Lerman), or by appointment
- **Text: Horstmann – Big Java (Second Edition) –**

11

## Academic honesty

- You may collaborate on understanding lectures, labs, text, tutorials, problem statements.
- You may discuss the design of your program: options for classes, method signatures.
- You must then write your Java code yourself.
- You may get help from students while writing your programs only by:
  - Asking them to point out an error, but not to fix it for you.
  - Explaining Java syntax to you. Use a different example than the program you're writing.
- You can get help from TAs, instructors when writing your program

12

## **Your Responsibilities for Loaner Laptops**

- **Practice “Safe Computing”**
  - Promiscuous use requires care
- **Provide good “care and feeding” of your laptop**
- **Return the computer at the end of the semester in good condition**
- **If the unthinkable happens...**
  - **Contact Campus Police for theft reporting**
  - **Notify your instructor/course technical contact immediately**

13

## **Mutual Responsibilities**

- **Backup**
  - **Use SecureFX file transfer utility to copy things to your personal MIT server locker**
  - ***You need to use it***
- **Recovery**
  - **Worst case scenario: machine re-imaged by IST (Information Systems and Technology) and restored to original working state**
  - **You load your data from your personal MIT server locker**

14

## **Course Outline**

**Course has 8 major units:**

- **Objects and Java**
- **Program structure**
- **Graphical user interfaces**
- **Numerical Methods**
- **Data Structures**
- **Java Input and Output**
- **Searching and Sorting**
- **Threads and the Web**

15

## **Class 1: Course Introduction and Overview of Java**

- **Java's history and goals**
- **What exactly is Java?**
- **Some key concepts in Java**
- **Some simple Java programs**

16

## **Java's History**

- **Java started as a Sun Microsystems research project to redesign C++**
- **Oak was going to be a C-- (C++ with dangerous features removed)**
- **Intended for consumer electronics, especially the early 90's interest in set-top boxes**

17

## **Java's History continued**

- **Then the WWW happened, and Oak became Java**
- **The functional requirements of the WWW serendipitously matched those of the interactive video market, which never developed**

18

## **Traditional Computing**

**Partitioning of functionality into:**

- **Operating system**
- **Programming languages**
- **Windowing systems**
- **Applications services (e.g. databases)**

19

## **Aspects of Traditional Computing**

- **Executable programs are specific to a hardware processor architecture and operating system.**
- **Applications typically preloaded onto computer with execution initiated by user.**
- **Client computers, servers, handheld devices separate environments**

20

## **Web Changed Everything**

- **Programs downloaded on demand from Web pages to client computers**
- **Client programs get extensive array of services**
- **Graphical user interface and event driven software the rule**

21

## **Java's Design Goals**

- **Safe, so you can trust application code downloaded over the WWW**
- **Portable, so you can develop on one system but run on another**
- **Distributed, so a "thin" client can take advantage of network services**
- **Scaleable, to build real applications based on extensive pre-existing class libraries**

22

## Some Features of Java

- Java is an entirely object oriented language. All programs involve objects.
- Java programs compile into a platform-independent machine code
- Java programs execute within the Java Virtual Machine
- An extensive collection of Java “packages” provide huge variety of solutions as leverage

23

## Object-oriented programming

- Objects are things ('entities') that have state (data fields) and behaviors (methods, functions)
  - They are a way of organizing large programs into understandable, maintainable, reusable pieces
  - Your programs, except for homework 1, will be a set of objects interacting with one another to produce the desired results
  - Examples will be pipes with fluid flows, bus routes in bus networks, elevators in elevator banks, polynomials, robots and stretch wrap devices, dictionaries of misspelled words, ...
- Classes are patterns from which objects are made

24

## Object-oriented programming

- Objects communicate by passing messages
  - They invoke behaviors (methods) and pass parameters (data) in messages
- Objects encapsulate or hide information
  - Details of one object are hidden from other objects, so their details need not be known
- “main method” often launches objects and does little else

25

## Object-oriented programming

- Objects are *extensible* through inheritance mechanisms
  - Children have parent’s traits (state and behavior) and can modify or add traits
  - Objects can dynamically invoke objects that didn’t exist (weren’t written yet) when the invoker was written. This, and other object concepts, promote code re-use.

26

## Developing a Java program

- Read the homework and understand it.
  - If you don't know what you have to do, you won't be able to do it.
- Sketch out a design: objects, state, behavior.
  - Decide how to approach the problem
  - Sketch the approach, in words or pictures. Sketch in stages.
- Write the program in Java, using Eclipse
  - Create Java source code files
  - Write Java code using Eclipse editor
  - Write only as much as you think will compile at each stage (e.g., reading the input). Invoke Java compiler from Eclipse
  - Once one stage compiles, write and compile the next. Stage size will increase over the term.

27

## Developing a Java program

- Test, mostly by reading/reviewing code
  - Use the Eclipse debugger to test code
- Repeat the cycle again to pick up details

28

## Four types of Java programs

- Console applications – text only
- Applets – run on Web pages with limited capabilities for security
- Frame-Based Applications – full, “free standing” programs
- Servlets – run on Web servers

29

## Simple console application

```
public class Welcome1 {
    public static void main(String[] args)
        System.out.println("Welcome to the Course");
        int students= 240;
        int grads= 35;
        double pctGrads= (double) grads/students;
        System.out.println("Percent grads: " +
            pctGrads);
        System.exit(0);
    }
}
```

// Lecture slides will use compressed format with {}  
// Use more white space in your code  
// Lecture slides will omit System.exit(0); you must use it

30

## Some rules for Java

- Each Java class should be in a separate file with the extension .java
- The file name should be same as the class name of the source code
- It's easiest to keep all files for a single program in one folder/directory

31

## Sample GUI application

```
// GUI application opens its own window (frame) on the PC
import javax.swing.*;
import java.awt.*;

public class Welcome extends JFrame {
    // Creates new form (object) Welcome
    // main called when application starts
    public static void main(String args[] ) {
        Welcome app= new Welcome();
        app.setDefaultCloseOperation(EXIT_ON_CLOSE);
        app.setVisible(true);
    }

    public Welcome( ) { // Constructor-called on creation
        JLabel myLabel= new JLabel ("Welcome to the course");
        setSize(300,200);
        Container conPane= getContentPane();
        conPane.add(myLabel);
    }
}
```

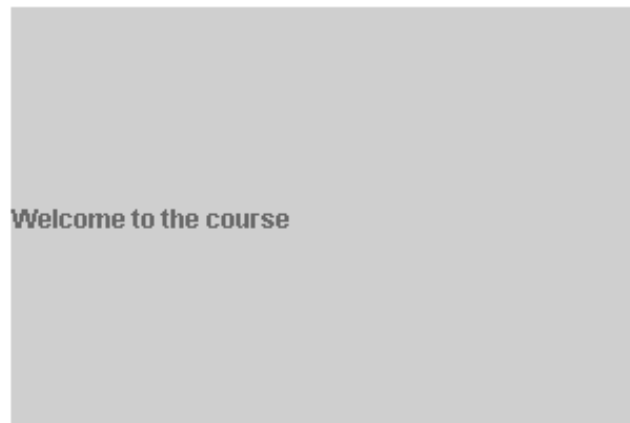
32

## A Simple Applet

```
import javax.swing.*;
public class CourseWelcome extends JApplet
{
    JLabel myLabel = new JLabel(
        "Welcome to the course");
    public void init()
    {
        getContentPane().add(myLabel);
    }
}
```

33

## Web page created by applet



34