Computer Science II

Professor Tom Ellman
Lecture 1

Goals of this Course

• Examines object-oriented programming, data structures and algorithms.
• By writing and analyzing computer programs.
• In Java, an object-oriented programming language.

Prerequisite

• Computer Science I (CMPU-101).
• Otherwise talk to Prof. Ellman to see if this is the right course for you.
Class Wiki
https://www.cs.vassar.edu/courses/cs102-201401-51/top

- Overview of the course.
- Schedule of lecture topics.
- Lab and homework assignments.
- Links to on-line textbooks.
- Summary of grading policy.
- Professor Ellman’s lecture notes.

Contacting Professor Ellman

- Just Google “Tom Ellman”.
- Office: 117 OLB
- Phone: 437-5991
- Email: thellman@vassar.edu

Office Hours

- Mondays: 12pm to 1pm.
- Tuesdays: 12pm to 1pm.
- Wednesdays: 12pm to 1pm.
- Thursdays: 12pm to 1pm.
Reading

- Oracle, The Java Tutorial (http://docs.oracle.com/javase/tutorial/).

Labs

- One lab each week.
- Fridays: 1:30pm-2:45pm.
- OLB 105.

Homework

- One programming assignment each week.
- Absolutely essential for learning the course material.
- Try not to fall behind, since it can be difficult to catch up.
Grading

• Participation: 10%
• Labs: 10%
• Assignments: 30%
• Midterm Exam: 20%
• Final Exam: 30%

Deadlines and Lateness

• Assignments are due at the start of class on the date specified.

• Late assignments will be accepted with 10% penalty, but only until the start of the next class.

Examinations

• Open book and open notes.

• Cumulative, from start of the semester.

• Exam dates shown on the class web page.
Academic Integrity

- You may discuss general ideas with classmates.
- You must do each programming assignment entirely by yourself.
- You may not discuss or share programs with other students.
- Vassar regulations require the professor to report suspected violations of academic integrity to the Dean of Studies.
- Read the “Originality and Attribution” pamphlet.

Java

- Why was Java invented?
  - To make programs portable via the internet.
- Why do we use Java?
  - It has all the main ingredients of object-oriented programming.
  - With a minimum of troublesome details.

Object-Oriented Programming

- Classes and Instances.
- Messages and Methods.
- Data Encapsulation.
- Inheritance.
- Polymorphism.
- Collections.
- Threads.
- Event Driven Programming.
Classes and Instances

- A **class** describes a variety of objects.
  - Specifies properties of all objects in the class.
  - Indicates ways in which objects in the class may differ from each other.
- An **instance** is an individual object.
  - Sharing the common properties of the class.
  - Manifesting properties that distinguish it from other objects in the class.

| Skin Color: __Brown__ | Shirt Color: __________ | Pants Color: __________ | Shoe Size: __________ | Shoe Color: __Black__ |
Java Development Process

- Programmer prepares Java source code.
- Programmer stores source code in file “PaperDoll.java”.
- Compiler translates source code into Java byte code.
- Compiler stores byte code in file “PaperDoll.class”.
- Compiled code runs on the Java Virtual Machine.
Java Development Process

- Programmer prepares Java source code.
- Compiler translates source code into Java byte code.
- Compiler stores byte code in file “PaperDoll.jar”.
- Compiled code runs on the Java Virtual Machine.

NetBeans

- Integrated Java Development Environment.
- Editor, Compiler, Debugger, and more.
- Free version available for Windows, Mac and Linux.
- See link to download site on our class web page.
- Be sure you have latest Java JRE and JDK.

HelloWorld.java

```java
package helloworld;
/**
 * The HelloWorld class implements an application that simply displays "Hello World!" to the standard output.
 */
public class HelloWorld
{
    public static void main(String[] args)
    {
        // Print out a greeting message.
        System.out.println("Hello World!");
    }
}
```
HelloWorld.java

Java Application Programs

• A file called “PaperDoll.java” holds a definition of a class called “PaperDoll”.
• The PaperDoll class has a method called “main” that is declared exactly as follows:

  ```java
  public static void main(String[] args) { ... }
  ```

• When the user runs the program PaperDoll.class, Java invokes the main method of the PaperDoll class.

ACM Student Package

• Library of Java classes that simplify input, output and interaction with users.

• Intended for use by students learning Java programming.

• Available on class wiki.
package easyinteraction;
import acm.program.*;
public class EasyInteraction extends ConsoleProgram {
    public void run()
            {
        println("Welcome to CMPU102: Topics in Computer Science");
        String firstName = readLine("Please enter your first name: ");
        String lastName = readLine("Please enter your last name: ");
        println("Hello, " + firstName + " + lastName + ".");
        println("It's a pleasure to have you with us!");
    }
    public static void main(String[] args) {
        new EasyInteraction().start(args);
    }
}

Make the ACM student package available to this program.

Give this program the capability of reading from and writing to a console window.

Display a string in the console window.

Read strings from the console and store in variables.

Concatenate strings and display in console window.

Program Interaction
package easydialog;
import acm.program.*;

public class EasyDialog extends DialogProgram {
    static final int currentYear = 2012;
    public void run() {
        String name = readLine("What is your name?");
        int age = readInt("How old are you?");
        int birthYear = currentYear - age;
        int altYear = birthYear - 1;
        print("Well, " + name + ", you were born in " + birthYear);
        println(" or else you were born in " + altYear + ");
    }
    public static void main(String[] args) {
        new EasyDialog().start(args);
    }
}

package easydrawing;
import acm.program.*;
import acm.graphics.*;
import java.awt.*;

public class EasyDrawing extends GraphicsProgram {
    public void run() {
        GRect base = new GRect(250, 250, 250, 200);
        base.setFilled(false);
        base.setColor(Color.BLUE);
        add(base);
        GLine roofLeft = new GLine(250, 250, 375, 150);
        roofLeft.setColor(Color.RED);
        add(roofLeft);
        GLine roofRight = new GLine(500, 250, 375, 150);
        roofRight.setColor(Color.RED);
        add(roofRight);
    }
    public static void main(String[] args) {
        new EasyDrawing().start(args);
    }
}
package multidrawing;

import acm.program.*;
import acm.graphics.*;
import java.awt.*;

class RedHouse extends GraphicsProgram { // ... See next slide ... }
class BlueHouse extends GraphicsProgram { // ... See next slide ... }

class MultiDrawing {
  public static void main(String[] args) {
    new RedHouse().start(args);
    new BlueHouse().start(args);
  }
}

class RedHouse extends GraphicsProgram {
  public void run() {
    GRect base = new GRect(250, 200, 250, 200);
    base.setFilled(true);
    base.setColor(Color.RED);
    add(base);

    GPolygon roof = new GPolygon();
    roof.addVertex(250, 200);
    roof.addVertex(375, 100);
    roof.addVertex(500, 200);
    roof.setFilled(true);
    roof.setColor(Color.BLACK);
    add(roof);

    GLabel label = new GLabel("Red House", 250, 450);
    add(label);
  }
}

class BlueHouse extends GraphicsProgram {
  public void run() {
    GRect base = new GRect(250, 200, 250, 200);
    base.setFilled(true);
    base.setColor(Color.BLUE);
    add(base);

    GPolygon roof = new GPolygon();
    roof.addVertex(250, 200);
    roof.addVertex(375, 100);
    roof.addVertex(500, 200);
    roof.setFilled(true);
    roof.setColor(Color.BLACK);
    add(roof);

    GLabel label = new GLabel("Blue House", 250, 450);
    add(label);
  }
}
class BlueHouse extends GraphicsProgram {
    public void run() {
        GRect base = new GRect(250, 200, 250, 200);
        base.setFilled(true);
        base.setColor(Color.BLUE);
        add(base);
        GPolygon roof = new GPolygon();
        roof.addVertex(250,200);
        roof.addVertex(375,100);
        roof.addVertex(500,200);
        roof.setFilled(true);
        roof.setColor(Color.BLACK);
        add(roof);
        GLabel label = new GLabel("Blue House",250,450);
        add(label);
    }
}