Introduction to data structures, algorithms, and object-oriented programming
Course Description

Computer Science II covers:
- fundamental data structuring concepts, including abstract data types (e.g., stacks, queues, lists, trees)
- fundamental algorithms (e.g., searching, sorting)
- introduction to analysis of algorithm complexity
- abstraction and inheritance in Java

The prerequisite for this course is CMPU 101. See your professor if you have not taken 101.
Textbook & Resources

- On-line text:
  
  *Introduction to Programming Using Java, version 7, David J. Eck*  
  (http://math.hws.edu/javanotes.)

Other Resources:
Java Documentation:
  
  http://docs.oracle.com/javase/7/docs/api/

Java Tutorial:
  
  http://docs.oracle.com/javase/tutorial/

ACM Student Package Documentation:
  
  http://jtf.acm.org/javadoc/student/
Labs

- One lab each week.
- Absolutely essential for learning the course material.
- Mondays: 4:35-5:50 pm.
- Sanders Physics 309.
• Approximately one programming assignment each week, except when tests are scheduled.
• Absolutely essential for learning the course material.
• Try not to fall behind, since it can be difficult to catch up.
Course Requirements

Students are required to:

1. complete all programming assignments over the course of the semester,

2. take 2 midterms and a final exam, and

3. complete all weekly laboratory assignments.
Examinations

- Two mid-term exams.
- One final exam.
- All exams:
  - Open book and open notes.
  - Cumulative, from start of the semester.
  - Done on-line in NetBeans.
  - Completed exam submitted to Moodle submission link.
Collaboration

• You may collaborate on labs. Such team work is encouraged in lab.

• You may NOT collaborate on homework or exams.
Assignment Lateness Policy

All programming assignments are due at midnight on the date specified.

Assignments will be worth 20 points.

Late assignments will receive a 1 point penalty for every day they are late unless you have a valid excuse from health services or the dean of studies. Valid retroactive excuses should be brought to my attention by coming to my office hours, not by e-mail.
Non-official Excuses

Valid excuses must come through
1. Baldwin Health Services
2. Dean of Studies office.

If you feel you need to hand an assignment in late, but you don't have a valid excuse from one of the above sources, make arrangements prior to the due date by visiting your instructor in person and getting permission in writing. A penalty of 1 point per day may still apply, depending on the circumstances.
Final grades will be computed as follows:

- 20% Assignments
- 25% Midterm exam I
- 25% Midterm exam II
- 25% Final exam (regularly scheduled time)
- 5% Lab and Lecture participation. Attendance is required and absences may negatively influence your final grade.
Lab and Homework Feedback

• Labs are not graded but attendance is mandatory. Your professor or a coach will need to see you (in person) demonstrate the completed lab before you get credit for attending.

• Homework is graded with one chance to correct your code for re-grading after initial feedback.

• You are strongly encouraged to visit me during my office hours for feedback on any of your course work.
All of your code must be original, written by you.

Any copying (electronic or otherwise) of another person's code violates academic integrity.

*Any student(s) suspected of going beyond the limits of academic integrity will be referred to the Dean of Students and the Academic Panel.*
Course Web Page

• All information about the course, syllabus, assignments, schedule, etc. is on our course Wiki page (*Check it frequently*):

• You will use our course Moodle page for submitting programs and labs.
Office Hours & Contact Information

**Office hours:**  
Mon, 10:15 am to 12:30 pm  
Mon, 2:30 to 3:30 pm  
Tue & Thu, 1:30 to 3:30 pm

**Office:**  
SP 306

**E-Mail:**  
jewalter@vassar.edu

**Office phone:**  
ext. 7449  
(leave message if no answer)

Reading assignment for 1st week: Chapters 1 and 2 of book at [http://math.hws.edu/javanotes](http://math.hws.edu/javanotes).
Academic accommodations are available for students with disabilities who are registered with the Office of Disability and Support Services. Students in need of disability accommodations should schedule an appointment with Professor Walter early in the semester to discuss any accommodations for this course that have been approved by the Office of Disability and Support Services, as indicated in your DSS accommodation letter.
Classroom Etiquette

Students are expected to attend all lectures, with exceptions permitted in case of excused illness and family emergencies. Lectures will begin on time. Students are expected to arrive on time. Professor Walter will make every effort to end each lecture on time as well. Students should not talk to each other during lectures. A student who wishes to ask a question should raise his/her hand and wait to be recognized. Students’ cell phones should be turned off during lectures.
Racket vs Java

Racket and other *functional* languages are designed to:

♦ organize algorithms as libraries of interacting *functions*
♦ use so-called “non-destructive programming” to allow for ease of testing

Java and other *object-oriented* languages are designed to:

♦ organize algorithms as libraries of interacting *methods*
♦ organize algorithms as collections of interacting *objects* with changeable state
♦ create *reusable, portable* code with a *hierarchical class* structure.
Java

• Why was Java created?
  – To make programs portable via the internet.

• Why do we use Java?
  – It has all the main ingredients of object-oriented programming (OOP).
  – It is one of the most common languages used in software engineering.
OOP Parts and Principles

• Classes and Instances
• Methods and Fields
• Packages
• Data Encapsulation
• Inheritance
• Polymorphism
• Event Driven Programming and Threads
• A *class* describes the contents of objects.
  – Specifies properties of all objects made from the class (shared or not shared).
  – Indicates ways in which objects in the class may differ from each other or be similar.

• An *instance* is an individual object.
  – Shares the common properties of the class.
  – Has properties that distinguish it from other objects in the class.
STEP 1:

- Create a file containing a *class* definition.
  - main body of program starts “public class ClassName”
  - all code is written within class body, delimited by braces.
  - the prefix of the file name must be the same as class name inside the file.

*NOTE:*

- Several classes may combine to form a complete program (called an *application*) but at least one class per application must contain the *main* method (execution always starts there).
HelloWorld.java

```java
/**
 * The HelloWorld class implements an application that
 * simply displays "Hello World!" to the standard output.
 */

class HelloWorld {
    // starting point for execution
    public static void main(String[] args) {
        // Print out a greeting message.
        System.out.println("Hello World!");
    }
}
```
STEP 2:

• Compile the program (type javac at command line, NetBeans has automatic compiling as a program is typed):
  – translates code into Java Byte Code
  – Byte code is stored in file with suffix “.class”

STEP 3:

• Run the byte code interpreter (java at command line) or press the Run icon in Netbeans.
Choosing names

• Class names must begin with a capital letter, and any subsequent words in the name are also capitalized (so-called “camel hump notation”)

Ex: A class to calculate scores for exam 3 might be called ExamThree.java or Exam3.java

A class to simulate a slot machine might be called SlotMachine.java or OneArmedBandit.java

Characters allowed in Java identifiers are letters, numbers, underscores and $. Class names must start with a capital letter. Variable and method identifiers follow same rules but start with a lowercase letter.
Class
Man

Instances of Man type

skinColor: __Brown__
shirtColor: _________
pantsColor: _________
shoeSize: _________
shoeColor: ___Black___
Java Development Process

Man.java

Java Compiler

Man.class

Java Virtual Machine

Computer

Input → Output
Java Development Process

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

- Man.java
- Helper1.java
- HelperN.java

Java Compiler

.class files

Java Virtual Machine

Computer

Input → .class files → Java Virtual Machine → Computer → .class files

Output
Java Development Process

• Programmer prepares Java source code in several classes that belong to the same package.

• Programmer stores source code in files: “Man.java”, “Helper1.java” … “HelperN.java”.

• Compiler translates source code into Java byte code: Man.class, Helper1.class”,…,HelperN.class

• Compiled code runs on the Java Virtual Machine.
Java Application Programs

• A file called “HelloWorld.java” holds a definition of a class called “HelloWorld”.

• The HelloWorld 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 HelloWorld.class, Java invokes the main method of the HelloWorld class.
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!");
    }
}
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!");
    }

}
DrRacket or DrScheme were IDEs you used to create and run your programs in 101.

There are many different IDEs for Java. We will use the NetBeans IDE. NetBeans is free to download and it comes packaged with the necessary Java Developer’s Kit and the JVM.

If you don’t have a CS account, see Jerry, our system administrator (SP307) and let your professor know.