CS133: Developing Programming Principles

Lecture 1

Introduction and Overview

Course Personnel

Instructor:

Email — _____@uwaterloo.ca
Office — DC ____
Phone —x_____
Hours — TBA, or by appointment, or drop in

Tutor:

Office — MC 4065 (CS Consulting Centre)
Email — cs133@student.cs.uwaterloo.ca
Hours — Refer to the course web page

Electronic resources

- **Web Page** -

  http://www.student.cs.uwaterloo.ca/~cs133

  - Check this **daily** for announcements and corrections
  - Assignments, lab exercises, and other important course information is posted on this page
  - Also: links and resources (e.g., Java API)

References

Required Texts

- *Java: An Introduction to Problem Solving & Programming, 4th ed.*, by Walter Savitch, Prentice-Hall (**Note:** you will need the 4th edition (2005), since it uses Java 1.5)

- *Course notes* (these contain all the slides)

Mark breakdown

- Assignments 15%
- Lab Assignments 15%
- Midterm Exam 20%
- Final Exam 50%

Assignments: Total of 5. Due roughly every other week.
Labs: Total of 10. Due weekly.
Midterm Date: Monday, October 22, 7 - 9 PM.
Final Exam date: *To be determined (December)*

You must pass the weighted average of the midterm and final exam in order to pass the course.

Policies – academic discipline

This reflects Math Faculty policy, so it applies to all of your Math Faculty courses. Individual courses may have more specific policies.

- Assignments are to be completed on your own, except as otherwise specified (usually group assignments). For example:
  - You may not use anyone else's solution
  - You may not discuss answers in detail with anyone
  - You may not copy anything from anywhere
- Standard penalty is a grade of -100% on the component (assignment or exam) in question
  - If the component is worth less than 5%, the standard penalty is an adjustment of -5% applied to your final course grade
  - If the component is worth more than 25% of the final grade, you cannot pass the course after the corresponding penalty
- Typical penalty for a second offence is suspension for a term
  - Plagiarism and exam cheating reduce the value of every other UW degree by casting doubt on the legitimacy of all students' achievements
  - Anybody who assists in the commission of an offence is also guilty
  - "Ignorance of the law is no excuse"
- University Policy 71 ("Student Academic Discipline Policy") contains relevant information and is available from the Web site of the University Secretariat at http://www.adm.uwaterloo.ca/infosec/

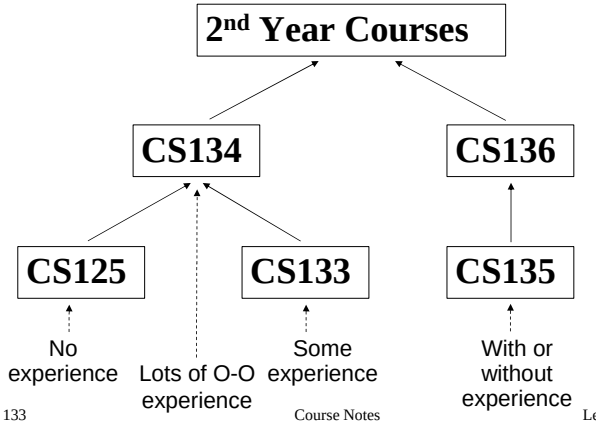
Course objectives

- Fundamental concepts of computer programming and computer science
- How to write small object-oriented programs in Java
- An introduction to graphical user interface (GUI) components

Course outline

- Introduction (1 hour)
- Review (5 hours)
- Basic Object-Oriented Concepts (12 hrs)
- Arrays (4 hours)
- Files (4 hours)
- Advanced O-O Concepts (4 hours)
- Graphical User Interfaces (6 hours)

First year CS courses



Our first Java program

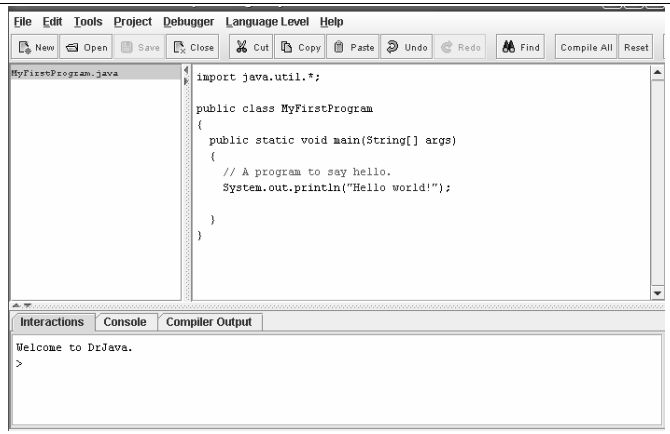
```
import java.util.*;

public class MyFirstProgram
{
  public static void main(String[] args)
  {
     // A program to say hello.
     System.out.println("Hello world!");
  }
}
```

Our first Java program continued

- Output:

```
Hello world!
```


DrJava



Compiling and running a program in DrJava

- Save your files
- Press "Compile All"
  - Fix compile-time errors
- Select the file with the **`main`** method (if you have many files)
- Select "Run document's main method"
  - Fix runtime errors
- Run your test cases

Types in Java

All data in Java is associated with a **type**.

The data's type determines:

- The syntax used to work with the data
- What operations are available
- How the data is stored in memory

Java type categories

Java organizes types into three categories:

- Primitive/base types
  - int, double, char, boolean, ...
- Object types (classes)
  - **`Scanner`**, **`Board`**, **`Rectangle`**, **`Student`**, ...
- "Oddball" types (objects that don't fit in)
  - **`String`**, arrays

Each category has its own rules and quirks

Primitive types

- **Integers**
  - examples: 3, -6, 12, 0, 19274
  - type **`int`**, **`byte`**, **`short`**, **`long`** in Java
- **Floating-point numbers**
  - examples: 10.2, -0.03, 1.0
  - type **`double`**, **`float`** in Java

Note: 10.0 is floating-point number but 10 is an integer.

More primitive types

- **Characters**
  - examples: 'a', 'x', '! '
  - type **`char`** in Java
- **Boolean**
  - examples: true, false
  - type **`boolean`** in Java

Table of primitives

Type Name	Kind of Value	Memory Used	Size Range
`byte`	*integer*	*1 byte*	-128 *to* 127
`short`	*integer*	*2 bytes*	-32768 *to* 32767
`int`	*integer*	*4 bytes*	-2147483648 to 2147483647
`long`	*integer*	*8 bytes*	-9223372036854775808 *to* 9223372036854775807
`float`	*floating-point number*	*4 bytes*	$\pm 3.40282347 \times 10^{+38}$ *to* $\pm 1.40239846 \times 10^{-45}$
`double`	*floating-point number*	*8 bytes*	$\pm 1.76769313486231570 \times 10^{+308}$ *to* $\pm 4.94065645841246544 \times 10^{-324}$
`char`	*single character (Unicode)*	*2 bytes*	*all Unicode characters*
`boolean`	`true` *or* `false`	*1 bit*	*not applicable*

From p. 53 of text

Declaring a variable

Type Variable1, Variable2, …;

Examples:

```
int styleNumber, numberofChecks;
char answer;
double amount, interestRate;
```

From p. 49 of text

Two ways to initialize variables

1.
- Declare a variable:
  `char quit;`
- Initialize variable:
  `quit = 'Q';`

2. Declare and initialize in one step:
   `char quit = 'Q';`

Operator precedence

- Same as in algebra:
  - **B**rackets
  - **E**xponents
  - **D**ivision
  - **M**ultiplication
  - **A**ddition
  - **S**ubtraction
- So `value = 3 * 4 + 7 – 4 / 2;`
  gives `value = 12 + 7 – 2 = 17;`

Special assignment operators

- Combine an arithmetic operator with the assignment operator

  Example:

  ```
  apples -= 2;
  ```

  is same as `apples = apples – 2;`

Increment and decrement operators

- **++** increases the value by 1
  Example: `count++;`
  is same as `count = count + 1;`

- **--** decreases the value by 1
  Example: `cars--;`
  is same as `cars = cars - 1;`

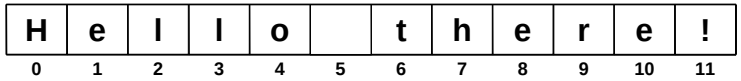
String

- Object holding a string of characters
  Example:
```
String message;
message = "This is it.";
```

- Initialized the same way as a primitive
  Example:
```
String greeting = "Hello.";
```

Character storage in `Strings` continued

```
String myString = "Hello there!";
```



Concatenation

- Joins two **`String`**s into one

  Example:

```
String firstPart = "Hello";
String secondPart = "there.";
String combined = firstPart + " " + secondPart;
```

  the value of combined is "**`Hello there.`**"

- For more info, see p. 77 of text

The + operator

- Notice:

```
int x = 10, y = 12, z;
String s1 = "hi", s2 = "bye", s3;
z = x + y;
s3 = s1 + s2;
```

- This is called **overloading**: the **operator** + does different things depending on its operands (specifically, the **type** of the operands).