Introduction

CMSC 202
Fall 2011
Instructors

- Mr. Ryan Bergeron
  - Lecture Section 01
    - Tues/Thu 1:00 – 2:15 am, Sondheim 111
  - Lecture Section 04
    - Tues/Thu 10:00 – 11:15 am, Sondheim 114
  - Lecture Section 10
    - Mon/Wed 8:30 – 9:45 am, Sondheim 110

- Ms. Susan Mitchell
  - Lecture Section 07
    - Mon/Wed 5:30 – 6:45 pm, Sondheim 204
What is CMSC 202?

• An introduction to object-oriented programming (OOP) and object-oriented design (OOD)
  • Uses the Java programming language
  • Uses the Eclipse integrated development environment (IDE)
• Strong emphasis on proper program design
• Course website:
  www.cs.umbc.edu/courses/undergraduate/202/fall11/
Procedural vs. OO Programming

**Procedural**
- Modular units: functions
- Program structure: hierarchical
- Data and operations **are not** bound to each other
- Examples:
  - C, Pascal, Basic, Python

**Object-Oriented (OO)**
- Modular units: objects
- Program structure: a graph
- Data and operations **are** bound to each other
- Examples:
  - Java, C++, Ruby

A Hierarchy of Functions

A Collection of Objects
What’s an Object?

• Must first define a **class**
  – A **data type** containing:
    • Attributes – make up the object’s “state”
    • Operations – define the object’s “behaviors”

<table>
<thead>
<tr>
<th>Bank Account</th>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>account number</td>
<td></td>
<td>sequence of characters more?</td>
</tr>
<tr>
<td>owner’s name</td>
<td></td>
<td>compute length</td>
</tr>
<tr>
<td>balance</td>
<td></td>
<td>concatenate</td>
</tr>
<tr>
<td>interest rate</td>
<td></td>
<td>test for equality</td>
</tr>
<tr>
<td>more?</td>
<td></td>
<td>more?</td>
</tr>
<tr>
<td>deposit money</td>
<td></td>
<td>more?</td>
</tr>
<tr>
<td>withdraw money</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check balance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transfer money</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Attributes (state)</th>
<th>Operations (behaviors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence of characters more?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compute length concatenate test for equality more?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
So, an Object is...

- A particular **instance** of a class

### Accounts

- **Bergeron’s Account**
  - 12-345-6
  - Ryan Bergeron
  - $1,250.86
  - 1.5%

- **Frey’s Account**
  - 65-432-1
  - Dennis Frey
  - $5.50
  - 2.7%

- **Mitchell’s Account**
  - 43-261-5
  - Susan Mitchell
  - $825.50
  - 2.5%

For any of these accounts, one can...

- Deposit money
- Withdraw money
- Check the balance
- Transfer money
Why Java for 202?

• Popular modern OO language
• Wide industry usage
• Used in many types of applications
• Desirable features
  – Object-oriented
  – Portability (cross-platform)
  – Easy handling of dynamic variables
  – Garbage collection
  – Built-in GUI libraries
Java History

• Created by **Sun Microsystems** team led by **James Gosling** (1991)

• Originally designed for programming home appliances
  – Difficult task because appliances are controlled by a wide variety of computer processors
  – Writing a compiler (translation program) for each type of appliance processor would have been very costly
  – Solution: **two-step translation process**
    • Compile, then
    • Interpret
Interpreters, Compilers, and the JVM

Interpreted Languages (e.g. JavaScript, Perl, Ruby)

- Source code → Interpret
  - **Interpreter** translates code into binary and executes it
  - Small, easy to write
  - Interpreter is unique to each platform

Compiled Languages (e.g. C, C++)

- Source code → Compile → Binary code → Execute
  - **Compiler** is unique to each platform

Java

- Source code → Compile → Bytecode
  - **Bytecode** is platform independent
- Bytecode → Interpret
  - **Java Virtual Machine (JVM)** is unique to each platform
Compiling and Running C/C++

- C/C++ Code
  - Linux C/C++ compiler
    - Linux binary
      - Linux C/C++ linker
        - Linux executable
      - Project Library for Linux
  - Windows C/C++ compiler
    - Windows binary
      - Windows C/C++ linker
        - Windows executable
      - Project Library for Windows
Compiling and Running Java

1. **Java Code**
   - `Hello.java`
   - `javac Hello.java` (Java compiler)

2. **Java Bytecode**
   - `Hello.class`

3. **JRE for Linux**
   - `java Hello`
   - Java interpreter (JVM) translates bytecode to machine code in JRE

4. **JRE for Windows**
   - `java Hello`
   - Java interpreter (JVM) translates bytecode to machine code in JRE
Java Terminology

• Java acronyms are plentiful and confusing. Here are the basics.

  – JVM – Java Virtual Machine
    • Translates Java bytecode to machine code
  – API – Application Programming Interface
    • The classes/methods/constants provided by libraries
  – JRE – Java Runtime Environment
    • The JVM and the Java API together
  – JDK (formerly SDK) – Java Development Kit
    • JRE + tools (compiler, debugger) for developing Java applications
    • The given edition of the JRE – standard being the most common
    • There are other versions that are tailored toward mobile devices and web environments

• To learn more about JDK, JRE, etc, visit:
Java SE Versions

• Current version of Java: Java 7, also known as Java 1.7 or Java 1.7.0

• Previous version: Java 6, also known as Java 1.6, Java 1.6.0 or “Java 2 SE Version 6”
  – This is the version running on GL servers

• To learn more about Java version naming, see: http://java.sun.com/javase/namechange.html
Python vs. Java

• Python

```python
print "Hello, world"
quotient = 3 / 4
if quotient == 0:
    print "3/4 == 0",
    print "in Python"
else:
    print "3/4 != 0"
```

Things to note:
• Everything has to be in some class
• We need a “main()”
• Statements end with “;”
• Variables must be declared
• “if/else” syntax different
• Statement blocks demarcated by “{...}”
• Comments are different
• Much that is similar

• Java

```java
public class Hello {
    public static void main(String[] args) {
        int quotient;
        System.out.println("Hello, world");
        quotient = 3 / 4;
        if (quotient == 0) {
            System.out.print("3/4 == 0");
            System.out.println(" in Java");
        } else {
            System.out.println("3/4 != 0");
        }
    }
}
```
The Eclipse IDE

• An integrated development environment (IDE) for writing Java programs. Contains (minimally):
  – Editor
  – Debugger
  – Java compiler
  – Java JVM

• Free (open source) download for Windows/Linux/Mac
  – See course “Resources” page on the CMSC 202 website

• Available in all OIT labs around campus
  – We’ll show you more in Lab 1
Eclipse IDE Screenshot