CMSC 202 Introduction

CMSC 202
Fall 2014

Instructors & Lecture Sections

- Dr. Christopher Marron
  - Section 01: M/W 8:30—9:45 AM, Administration 101
- Mr. John Park
  - Section 06: Tu/Th 1:00—2:15 PM, ITE 104
- Mr. James Kukla
  - Section 06: M/W 5:30—6:45 PM, Sondheim 101

What is CMSC 202?

- An introduction to
  - *Object-oriented programming* (OOP) and *object-oriented design* (OOD)
  - Basic *software engineering* techniques

- Strong emphasis on *proper program design* and *maintainability*

- Tools
  - C++ programming language, GCC (Gnu C Compiler)
  - Linux (GL system)
  - [Optionally: Eclipse integrated development environment (IDE)]

Course Web Site

(cs.umbc.edu/202/fall14 will also work)

8/26/2014   4

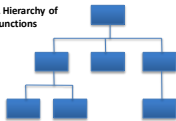
Procedural vs. OO Programming

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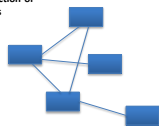- Examples:
  - C, Pascal, Basic, Python

Object-Oriented (OO)
- Modular units: objects
- Program structure: a graph
- Data and operations <u>are</u> bound to each other
- Examples:
  - C++, Java, Python (huh?!)

A Hierarchy of Functions

A Collection of Objects

8/26/2014   5

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*

Bank Account			String
account number owner's name balance interest rate more?	←— Type —→		sequence of characters more?
	←— Attributes (state) →→		
deposit money withdraw money check balance transfer money more?	←— Operations (behaviors) —→		compute length concatenate test for equality more?
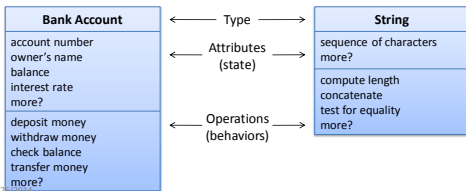
8/26/2014   6

So, an Object is…

- a particular *instance* of a class

Marron's Account	Kukla's Account	Park's Account
12-345-6 Chris Marron $1,250.86 1.5%	65-432-1 James Kukla $5.50 2.7%	43-261-5 John Park $825.50 2.5%

For any of these accounts, one can…
- Deposit money
- Withdraw money
- Check the balance
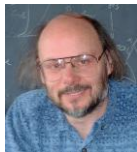- Transfer money

Why C++ for 202?

- Popular modern OO language
- Wide industry usage
- Used in many types of applications
- Desirable features
  – Object-oriented
  – Portable (not as much as Java, but fairly so)
  – Widespread use in industry
  – Efficient
  – Retains much of its C origins

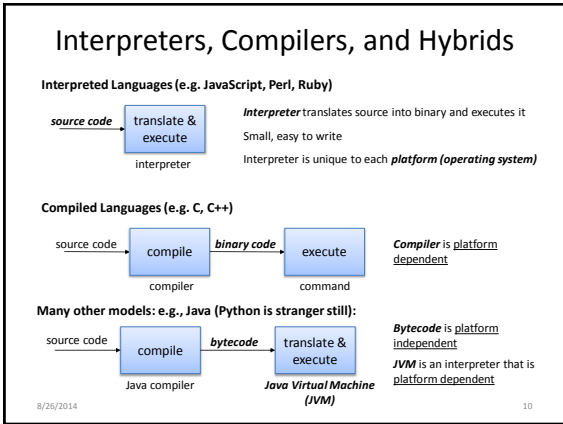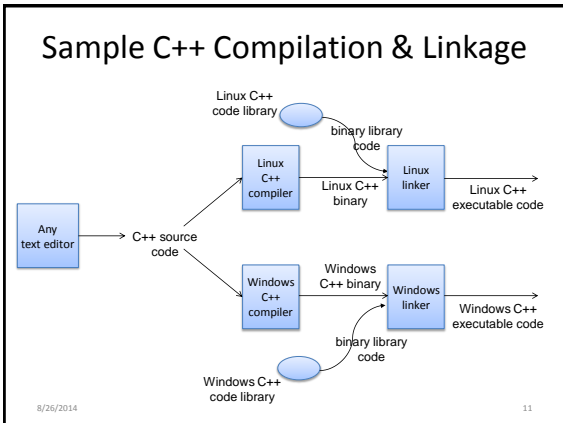## Some C++ Background

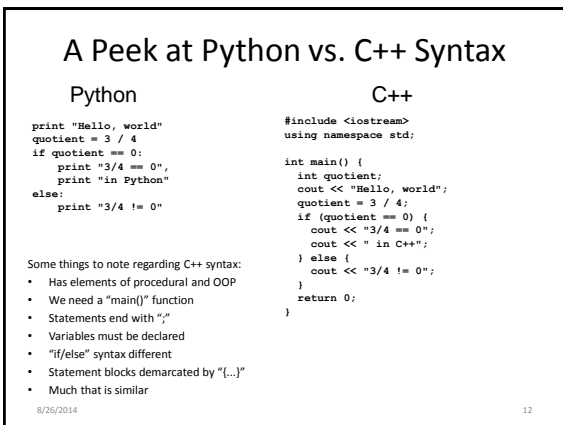Bjarne Stroustrup
(image from home page)

- Created in 1979 by Bjarne Stroustrup of Bell Labs (home of UNIX and C)
- added object-oriented features to C (called "C with Classes" at first)
- Shortly renamed to "C++" in honor of C's auto-increment operator.
- Later standardized with several ISO specifications (most recently 2011)
- Greatly influenced Java development (1991)

Interpreters, Compilers, and Hybrids

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

source code → [translate & execute] (interpreter)

Interpreter translates source into binary and executes it

Small, easy to write

Interpreter is unique to each *platform (operating system)*

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

source code → [compile] (compiler) → *binary code* → [execute] (command)

Compiler is platform dependent

Many other models: e.g., Java (Python is stranger still):

source code → [compile] (Java compiler) → *bytecode* → [translate & execute] (*Java Virtual Machine (JVM)*)

Bytecode is platform independent

JVM is an interpreter that is platform dependent

8/26/2014   10

Sample C++ Compilation & Linkage

8/26/2014   11

A Peek at Python vs. C++ Syntax

Python

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

C++

```
#include <iostream>
using namespace std;

int main() {
  int quotient;
  cout << "Hello, world";
  quotient = 3 / 4;
  if (quotient == 0) {
    cout << "3/4 == 0";
    cout << " in C++";
  } else {
    cout << "3/4 != 0";
  }
  return 0;
}
```

Some things to note regarding C++ syntax:
- Has elements of procedural and OOP
- We need a "main()" function
- Statements end with ";"
- Variables must be declared
- "if/else" syntax different
- Statement blocks demarcated by "{...}"
- Much that is similar

8/26/2014   12

Development Environment

- We will use the GL systems and GCC (GNU Compiler Collection) suite for development
- You will learn to be semi-literate in UNIX (Linux) and shell usage
- You should learn Emacs (important for later courses)
- We might provide instructions for using Eclipse, but support might be limited

8/26/2014   13

A Peek at the Eclipse IDE

- ***Integrated Development Environment*** (IDE) for writing programs in various languages. Contains:
  - text editor
  - Integration environment for:
    - debugger
    - C++ compiler
- Free download for Windows/Linux/Mac
  - See course "Resources" page on the CMSC 202 web site
- Available in all OIT labs around campus
  - We'll show you more in Lab 1

8/26/2014   14