CSC 207
Algorithms and Object-Oriented Design

Object Orientation with Java

I am indebted to my colleague, John David Stone, for assistance with this handout.
This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit
https://creativecommons.org/licenses/by-sa/4.0/
or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Introduction

Shervin Hajiamini
Visiting Computer Science Faculty
Before: PhD at Washington State University
Email address: hajiamini@grinnell.edu
Office location: Noyce 3824
Office Hours: Tuesdays and Fridays 3:00-5:00 pm or by appointment
Mentors: Mia Le and Garrett Wang

http://www.cs.grinnell.edu/~hajiamini/courses/CSC207/
- Syllabus
- Schedule of Topics
- Labs, assignments, and handouts
Notes from Syllabus

• CSC 207 learning objectives
  – Model a problem by dividing it into data and operations
  – Object-oriented model
    ▪ Separate implementation details from users
    ▪ Use Java programming language
    ▪ Develop Java-based programs for solving problems
Notes from Syllabus (cont.)

• Grading breakdown
  – Exercises (programming assignments): 70%
  – Class attendance and participation: 10%
  – Examination: 20%

• Hints to do well
  – Read assigned readings before the class
  – Complete labs as much as you can
  – Mentor session (2 hours per week)
  – Work on the Study Questions (important for the final examination)
Notes from Syllabus (cont.)

• Collaboration policy
  – Labs: pair up with another student
  – Programming assignments: discuss ideas and concepts with CSC 207 instructor, mentors, and peers in your section
    ▪ Programming assignments must be done individually.

• Academic honesty
  – Acknowledge sources that you get help from when working on the programming assignments
Object-oriented programming

• Emerged as a programming model in mid-1990s.

• Object
  – Data type
  – State
  – Operations (methods)

• Atomicity
  – Information hiding
  – Encapsulation

• Code reuse
Object-Oriented Programming (cont.)

• Inheritance
  – Extend the state and operations of an existing object

• Class
  – Similar to struct in C or record in Pascal
  – Members: methods and fields
  – Access fields by invoking methods on objects

• Library
  – Hierarchy of standard classes
  – Application Programming Interface (API)
Information Hiding

• Methods
  – Create an instance of class (object) and initialize it.
    • Dynamic memory (de) allocation
  – Perform specific operations, e.g. I/O.

• Declare fields as private members
  – Only methods of a class can access to fields
Static Members

• A class can have its own fields and methods.

• Declare these members with *static* keyword

• Private, static members are only accessible to the objects of the class.
“Hello, World!” program
Information Hiding (cont.)

**figure 3.1**
A complete declaration of an IntCell class

```java
1 // IntCell class
2 // int read() --> Returns the stored value
3 // void write( int x ) --> x is stored
4
5 public class IntCell
6 {
7    // Public methods
8    public int read() { return storedValue; }
9    public void write( int x ) { storedValue = x; }
10   
11    // Private internal data representation
12    private int storedValue;
13 }
```

**figure 3.3**
A simple test routine to show how IntCell objects are accessed

```java
1 // Exercise the IntCell class
2
3 public class TestIntCell
4 {
5    public static void main( String [ ] args )
6    {
7        IntCell m = new IntCell( );
8
9        m.write( 5 );
10        System.out.println( "Cell contents: " + m.read( ) );
11
12        m.storedValue = 0;
13    }
14 }
```