1. General Information

Course Title: CSCI-605 Advanced Object-Oriented Programming Concepts
Instructor: Hans-Peter Bischof
Office: 70-3005
Telephone: (585) 475-5568
Office Hours: [Tues|Thurs]day: 10am - noon, and by appointment
Course Home Page:
CS Home Page:
1.1. Topics

• 1 - Inheritance
• 1 - Generics
• 1 - Wildcards
• 2 - I/O: Files and Streams
• 2 - Threads
• 3 - Networking
• 3 - Collections
1.2. Grading

93-100% A
90-92% A-
87-89% B+
83-86% B
80-82% B-
77-79% C+
73-76% C
70-72% C-
60-69% D
Below 60% F

1.3. Tentative Schedule

Exam 1: jan/13/2016
Exam 2: jan/22/2016 - I might have to move this one

I will use concepts from the One part of each homework will use concepts not covered in class. I will point where you can find these materials. The first example is hw 1.[23].

1.4. Syllabus

1.5. Academic Honesty

— Academic Integrity
— You may help each other freely to complete homeworks as the purpose of the homeworks is to increase your understanding.
— This does not mean that someone else can do your homework for you. Any homework you submit must contain your significant intellectual contribution.
— The corollary is that you may not do someone else’s work for them either. A willing supplier of the material is as guilty of academic dishonesty as the receiver.
— Any help you receive from someone must be acknowledged in the work submitted. Failure to acknowledge the source of a significant idea or approach is considered plagiarism and not allowed.

Academic dishonesty is dealt with severely:
— You will receive a grade of F for the course.
— A note describing the details of your case will become part of your academic record.
— Repeated offenses or more serious violations may result in your being suspended or
— Violations of the Academic Integrity can also result in suspension, expulsion and even criminal charges.
1.6. Course Goals

- Intro into the language Java (1.7)
- Use of the major classes, like Collection Framework, RMI, Swing, etc.
- Introduction to design patterns.

This is not a “programming” course, per se. Programming is a means to an end, not an end in and of itself.
1.7. Web Resources

Example:

```java
public class HelloGoodbye {
    public static void main(String[] args) {
        System.out.println("Hello world");
        Runtime.getRuntime().addShutdownHook(
            new Thread() {
                public void run() {
                    System.out.println("Goodbye world");
                }
            });
        System.exit(0);
    }
}
```

Source Code: Src/1/HelloGoodbye.java

1.8. Texts

There is an almost infinite number of (not necessarily good) books about Java and even more about the World Wide Web. The following books I found quite useful, however.

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bruce Eckel</td>
<td>Thinking in Java</td>
<td>mindview.net/Books</td>
</tr>
<tr>
<td>Java Tutorial</td>
<td>javasun.com/docs/books/tutorial/index.html</td>
<td>java</td>
</tr>
<tr>
<td>Online Courses</td>
<td>javasun.com/developer/onlineTraining/ Online Courses</td>
<td></td>
</tr>
<tr>
<td>Mughal/Rasmussen</td>
<td>0-201-59614-8</td>
<td>A Programmers Guide to Certification</td>
</tr>
<tr>
<td>Kalin</td>
<td>0-13-019859-5</td>
<td>Object Oriented Programming in Java</td>
</tr>
<tr>
<td>Flanagan</td>
<td>1-56592-262-X</td>
<td>Java in a Nutshell (2nd Edition)</td>
</tr>
<tr>
<td>Flanagan</td>
<td>1-56592-371-5</td>
<td>Java Examples in a Nutshell</td>
</tr>
</tbody>
</table>

Useful is also The Java Language Specification:

The following books were written by the Java developers themselves and are relatively useful. Second editions are available or in the making, more or less face-lifted for Java 1.23:

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arnold/Gosling</td>
<td>0-201-31006-6</td>
<td>The Java Programming Language (2nd Ed.)</td>
</tr>
<tr>
<td>Campione/Walrath</td>
<td>0-201-31007-4</td>
<td>The Java Tutorial (2nd Ed.)</td>
</tr>
<tr>
<td>Chan</td>
<td>0-201-37967-8</td>
<td>The Java Developers Almanach (1998 Ed.)</td>
</tr>
<tr>
<td>Chan/Lee</td>
<td>0-201-63458-9</td>
<td>The Java Class Libraries: An Annotated Reference</td>
</tr>
<tr>
<td>Chan/Lee/Kramer</td>
<td>0-201-31002-3</td>
<td>The Java Class Libraries (2nd Ed.) Vol. 1</td>
</tr>
<tr>
<td>Chan/Lee</td>
<td>0-201-31003-1</td>
<td>The Java Class Libraries (2nd Ed.) Vol. 2</td>
</tr>
<tr>
<td>Gosling/Joy/Steele</td>
<td>0-201-63451-1</td>
<td>The Java Language Specification</td>
</tr>
<tr>
<td>Gosling/Yellin et al.</td>
<td>3-8273-1040-7</td>
<td>Java API Volume 1: Basic Packages</td>
</tr>
<tr>
<td>Gosling/Yellin et al.</td>
<td>3-8273-1084-9</td>
<td>Java API Volume 2: Window Toolkit and Applets</td>
</tr>
<tr>
<td>Hamilton/Cattell/Fisher</td>
<td>0-201-30995-5</td>
<td>JDBC -- Database Access with Java</td>
</tr>
<tr>
<td>Kanerva</td>
<td>0-201-63465-2</td>
<td>The Java FAQ: Frequently Asked Questions</td>
</tr>
<tr>
<td>Lea</td>
<td>0-201-69581-2</td>
<td>Concurrent Programming in Java</td>
</tr>
<tr>
<td>Liang/Sterns</td>
<td>0-201-32577-2</td>
<td>Java Native Interface</td>
</tr>
<tr>
<td>Lindholm/Yellin</td>
<td>0-201-63452-X</td>
<td>The Java Virtual Machine Specification</td>
</tr>
<tr>
<td>Sowizral/Rushforth et.al.</td>
<td>0-201-32576-4</td>
<td>The Java 3D API Specification</td>
</tr>
<tr>
<td>Sridharan</td>
<td>0-13-749136-0</td>
<td>Advanced Java Networkin</td>
</tr>
</tbody>
</table>

If you would like to take a Java exam to become a Sun Certified Programmer for the Java 2 Platform, here is an example:
1.9. Environment
Java 1.7 will be used

1.10. Homew orks
The solutions for the homeworks are in Java 1.7.
The homeworks are done in teams of two. The team has to meet with a grader. Each team members must be able to explain the solution to her/him. The grade for each project is based on the correctness, your explanation, and the quality of the code.

A solution must be submitted by each student for the first homework.
You can submit as often as you like, but only the last submission counts.
Signup sheets will be up at the graduate lab door by Wednesday. Be on time. You might receive 0 points for the homework if you are late.
1.11. Other Things

Course Organization

• cheating
• lectures - ask questions
• homework
  — Grader
  — Individual questions
1.12. Object-Oriented Ingredients

- Objects
- Class
- Methods
- Encapsulation
- Inheritance
- Polymorphism

1.13. Object

Objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process.

In between, each object is made into a generic class of object and even more generic classes are defined so that objects can share models and reuse the class definitions in their code.

Each object is an instance of a particular class or subclass with the class’s own methods or procedures and data variables. An object is what actually runs in the computer.

1.14. Class

A class consists of all objects with like state which know exactly the same methods, i.e., a class knows the structure of its objects and contains the methods to manipulate the structure. An object is called an instance of a class.

Given a class, one normally creates objects. Objects are created dynamically with the prefix operator new which in turn calls a constructor method to initialize the instance variables. Uninitialized instance variables are zeroed.

Methods mostly access the instance variables of the receiver. If methods return their receiving objects, method calls (messages) can be cascaded.

The class is one of the defining ideas of object-oriented programming. Among the important ideas about classes are:

1.15. Encapsulation

Encapsulation is the inclusion within a program object of all the resources need for the object to function — basically, the methods and the data.

Other objects adhere to use the object without having to be concerned with how the object accomplishes it. The idea is "don’t tell me how you do it; just do it.” An object can be thought of as a self-contained atom. The object interface consists of public methods and instance data.

1.16. Methods

A method is a programmed procedure that is defined as part of a class and included in any object of that class. A class (and thus an object) can have more than one method. A method in an object can only have access to the data known to that object, which ensures data integrity among the set of objects in an application. A method can be re-used in multiple objects.

- A class can have subclasses that can inherit all or some of the characteristics of the class. In relation to each subclass, the class becomes the superclass.
• Subclasses can also define their own methods and variables that are not part of their superclass. The structure of a class and its subclasses is called the class hierarchy.

Question: What is the difference between class and object

1.17. Inheritance
Inheritance is the concept that when a class of objects is defined, any subclass that is defined can inherit the definitions of one or more general classes.

This means for the programmer that an object in a subclass need not carry its own definition of data and methods that are generic to the class (or classes) of which it is a part.

This not only speeds up program development; it also ensures an inherent validity to the defined subclass object (what works and is consistent about the class will also work for the subclass)

1.18. Polymorphism
Polymorphism (from the Greek meaning "having multiple forms") is the characteristic of being able to assign a different meaning to a particular symbol or "operator" in different contexts.

1.19. Object-Oriented Programming Keywords
• Object
• Class
• Encapsulation
• Methods
• Encapsulation
• Methods
• Inheritance
• Polymorphism
• Reuse