CSE 219: Computer Science III

Systematic Program Design, Coding, and Testing
Course Description

- Development of the basic concepts and techniques from Computer Science I (CSE114) and II (CSE214) into practical programming skills that include a systematic approach to program design, coding, testing, and debugging.
- Application of these skills to the construction of robust programs of thousands of lines of source code.
- Use of programming environments and tools to aid in the software development process.

Prerequisite

I. C or higher grade in CSE 214
II. CSE or ECE major
Course Information

Lectures
● Monday and Friday, 1:00 PM - 2:20 PM @ 104 FREY HALL

Homework Assignments
● Everything must be submitted via Blackboard

Programming
● JDK 1.8 (if you have an older version, update NOW!)
● NetBeans or IntelliJ IDEA

Source Code Management
● Version control using git, with Bitbucket being used as the remote repository
  ○ https://bitbucket.org/

Discussion Board
● Piazza
  ○ https://piazza.com/stonybrook/fall2017/cse219
  ○ Access code: llysr
  ○ Sign up with your stonybrook email and please do NOT share with anyone outside this class
Course Information

Instructor

- Dr. Ritwik Banerjee (rbanerjee@cs.stonybrook.edu)
  206 New Computer Science Bldg.
- Office Hours:
  - Mon 11:00 am - 12:30 pm
  - Fri 2:30 pm - 4:00 pm

Course Page

  - Main page for both sections. Includes the overall schedule.
  - This is our section-specific page, and you can safely just use this page. Anything on the first “common” page is linked from this one.

Texts

(both available electronically from SBU library)

- **Head First Object Oriented Analysis and Design**
  Brett McLaughlin, Gary Pollice, and David West.
  Publisher: O'Reilly Media, Inc. 2006.
- **Head First Design Patterns**
  Eric Freeman, Elisabeth Robson, Bert Bates, and Kathy Sierra
  Publisher: O'Reilly Media, Inc. 2004.
Course Topics

● **Code design**
  ○ Decomposing large problems into smaller, reusable and interconnected components using simple interfaces, i.e., modular design.
  ○ Object oriented design.

● **Selecting appropriate data structures and algorithms.**
  ○ Unlike CSE 214, you will have to decide what to apply.

● **Learning to quickly understand and use code/libraries written by others.**

● **Learning to systematically test and debug large code.**

● **Learning to maintain a code repository during incremental project development using source code management.**
Course Outcomes

● The goals of this course, as per the department of computer science, are:
  ○ Ability to systematically design, code, debug and test programs of about two or three thousand lines of code.
  ○ Sensitivity to the issues of programming style and modularity and their relationship to the construction and evolution of robust software.
  ○ Knowledge of basic ideas and techniques of object-oriented programming.
  ○ Familiarity with the capabilities and use of programming tools such as syntax directed editors, debuggers, execution profilers, documentation generators, and revision-control systems.
The work

- **Five homework assignments** building towards the complete final project
- **Exams**: one midterm and one final exam
- **Final Project**
  - This is the sixth assignment, and the final submission is the culmination of all your hard work through the semester.
- **Grading Scheme**
  
<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Homeworks</td>
<td>25% (5% each)</td>
</tr>
<tr>
<td>Final Project</td>
<td>20%</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>25%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>25%</td>
</tr>
<tr>
<td>Recitation</td>
<td>5%</td>
</tr>
</tbody>
</table>

  ➔ Late submissions are not accepted for grading.
  ➔ Code that does not compile will not be accepted for grading.
Communication Policy

- Do not use any email address for the instructor other than the one provided on his home page (the same as provided in this lecture).
- Do not use any email address to communicate with a TA other than the one provided on the ones provided in the course web page.
- Clearly state the course in your email subject.
  - For example, "CSE 219: homework 3 doubt".
- Follow announcements carefully.
  - Emails asking for information already available in announcements will not receive a response.
Re/Grading Policy

● All homework and final project grading will be done by appointments.
  ○ Regrading later, or grading homework assignments by any other means, is not possible.

● The mid-term exam papers will be retained by the instructor at first.
  ○ Any request for regrading must be made during instructor office hours within a 2-week period from the day the grade was provided on Blackboard.
  ○ Mid-term papers, once returned (which will be done during instructor office hours), will no longer be considered for regrading.

● For the final exam, there will be a special office hour designated to resolve any grade queries or disputes. This will be announced after the final exam.
  ○ The final exam papers are not returned.
Academic Integrity

You may discuss the homework in this course with anyone you like, however each student's submission, including written material and coding, must be his or her own work, and only his or her own work.

Any evidence that written homework submissions or source code have been copied, shared, or transmitted in any way between students (this includes using source code downloaded from the Internet or written by others in previous semesters!) will be regarded as evidence of academic dishonesty. Additionally, any evidence of sharing of information or using unauthorized information during an examination will also be regarded as evidence of academic dishonesty.
Academic Integrity

Each student must pursue his or her academic goals honestly and be personally accountable for all submitted work. Representing another person's work as your own is always wrong. Faculty are required to report any suspected instances of academic dishonesty to the Academic Judiciary. Faculty in the Health Sciences Center (School of Health Technology & Management, Nursing, Social Welfare, Dental Medicine) and School of Medicine are required to follow their school-specific procedures. For more comprehensive information on academic integrity, including categories of academic dishonesty, please refer to the academic judiciary website:

- [http://www.stonybrook.edu/commcms/academic_integrity/index.html](http://www.stonybrook.edu/commcms/academic_integrity/index.html)
Disability Support

If you have a physical, psychological, medical or learning disability that may impact your course work, please contact Disability Support Services, ECC (Educational Communications Center) Building, room 128, (631) 632-6748. They will determine with you what accommodations, if any, are necessary and appropriate. All information and documentation is confidential. Students who require assistance during emergency evacuation are encouraged to discuss their needs with their professors and Disability Support Services. For procedures and information go to the following website: [http://www.stonybrook.edu/ehs/fire/disabilities](http://www.stonybrook.edu/ehs/fire/disabilities)
Critical Incident Management

Stony Brook University expects students to respect the rights, privileges, and property of other people. Faculty are required to report to the Office of Judicial Affairs any disruptive behavior that interrupts their ability to teach, compromises the safety of the learning environment, or inhibits students' ability to learn. Faculty in the HSC Schools and the School of Medicine are required to follow their school-specific procedures.
What is this course all about?

- As students, we learn coding with small “toy” problems, problems where the entire process is only about coding.
- But the world of software engineering is not like that. There,
  - we have a large amount of data
  - we must DESIGN first, and only then write code
  - we must know the details of programming paradigm (in our case, that’s OOP)
- In short, CSE 219 is the first step towards transforming your programming skills to that of a professional software engineer.
- To do that, we must learn about the software development lifecycle.
  - And to understand THAT, we need to ask a very important question!
What makes good quality software?

The salient points are

● Correctness
● Efficiency
● Ease-of-use (either end users, or by other programmers ... depending on what you’re building)
● Reliability
● Maintainability
● Modifiability and extensibility
● Scalability
How to achieve these goals?

Communication ... of different types

- As programs get bigger, the different parts begin to communicate in unforeseen (and usually bad) ways!
- As programming teams get larger, programmers begin to communicate in unforeseen (and usually bad) ways!
  - The most important “bad communication” is often a lack of complete and precise communication.

Image courtesy: Cornell University
The “software development lifecycle”

- As programs get larger, writing good software becomes harder because
  - Program complexity increases
  - Team complexity increases
- Under these conditions, how can we still write good software?
  - By following well-established processes
  - And by taking advantage of good tools already available.
  - Never reinvent the wheel!
The “software development lifecycle”

Some other aspects involve

- **Software integration**
  - To combine developed software into a single cohesive unit. This is typically done for larger projects.

- **Software maintenance**
  - Shown as the last step in the lifecycle here.
  - Usually continues even *after* the deployment of the final product.
  - Involves monitoring and updating.
The “software development lifecycle”

This is known as the **Waterfall Model** of software development.

- There are many variations of the basic model structure.

Some other models are

- Agile model
- Iterative model
- V model
- Spiral model
- etc.
Some typical jobs in the software industry

- **Programmers**
  - The most time-consuming job in software development.
  - Additionally, a programmer has to be able to design, test and debug complex software.

- **Non-programmers**
  - Designer
  - Database, Network, Security Administrator
  - Tester
  - Project Leader
  - Manager
  - Founder/CEO

*NOTE that designers and programmers on a project may not be the same people!*
Design ... then develop

● We will design all classes before coding.
  ○ Not easy to do!
  ○ We will use UML (unified modeling language) for software design.

● A system cannot be designed
  ○ unless you really understand the necessary technology!

● A design cannot be created
  ○ without testing the design first to identify any potential flaws or shortcomings.
Frameworks, or how to not reinvent the wheel

This is where we start talking about using available tools. In particular, we will focus on frameworks, which are collections of multiple classes that work together to allow customizable use of some technology.

- Groups of classes that form the basis for customization:
  - Cooperating classes for a particular technology (e.g., multimedia, the Web, databases).
  - Often used to build new applications, or even other frameworks!
  - E.g., what is a Java application framework for GUI development?
Frameworks, or how to not reinvent the wheel

This is where we start talking about using available tools.

- Focus on frameworks, which are collections of multiple classes that work together to allow customizable use of some technology.
- Groups of classes that form the basis for customization:
  - Cooperating classes for a particular technology (e.g., multimedia, the Web, databases).
  - Often used to build new applications, or even other frameworks!
  - E.g., what is a Java application framework for GUI development? **JavaFX**.
- Application using a framework:
Some common Java frameworks

- Spring MVC
- Grails
- Spark
- Vaadin

Framework developers must explain how to use them all together properly:
- API
- Commenting in code
- Tutorials

Frameworks may be open source or proprietary. Gaining the ability to make frameworks will make you a very powerful (and sought after!) developer.
Graphical User Interfaces (GUIs)

Why? Why? Why?

- “Ease of use” is an important criteria for good software, so why not a GUI?
- We, as users, use GUIs all the time!
Graphical User Interfaces (GUIs)

- A GUI does not automatically mean it’s a well-designed software. A GUI can be good or bad.
  - A well-designed GUI is more user-friendly and intuitive.
- JavaFX is a framework for building GUI applications with Java.
- There are other (older) frameworks too, like
  - Java Swing (`javax.swing.*`)
  - Java’s event programming libraries (`javax.awt.event.*; javax.swing.event.*`)
  - Java’s graphics programming libraries (`javax.awt.*; java.awt.geom.*`
Graphical User Interfaces (GUIs)

- A GUI is a reactive system. It responds to “events” and stays in a loop.

```
Construct GUI Components

Render GUI

Check for input

Respond to input
```
GUI example: a mouse click

- The OS recognizes a mouse click
  - Determines which window it was inside.
  - Notifies that program.

- That program runs in loop:
  - Checks input buffer filled by OS.
  - If it finds a mouse click:
    - Determines which component in the program.
    - If the click was on a relevant component.
    - Respond appropriately according to handler.

- This is an example of GUI behavior.

- But a GUI has another important aspect: it’s look!
GUI: “look” and “behavior”

<table>
<thead>
<tr>
<th>Look</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical appearance</td>
<td>Interactivity</td>
</tr>
<tr>
<td>Custom component design</td>
<td>Event-programmed response</td>
</tr>
<tr>
<td>Containment</td>
<td></td>
</tr>
<tr>
<td>○ A container is simply something that provides space for a component to be located.</td>
<td></td>
</tr>
<tr>
<td>Layout management</td>
<td></td>
</tr>
</tbody>
</table>
What does a GUI framework do? A JavaFX example.

```java
public class JavaFXApplication1 extends Application {

    @Override
    public void start(Stage primaryStage) {
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Example Window");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```
JavaFX vs. Swing vs. AWT

● When Java was introduced, the GUI classes were bundled in a library known as the Abstract Windows Toolkit (AWT).
  ○ AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects.
  ○ In addition, AWT is prone to platform-specific bugs.

● The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as Swing.
  ○ Swing components depend less on the target platform and use less of the native GUI resource.

● With Java 8, Swing is replaced by a completely new GUI platform: JavaFX.