Introduction and Overview

Introduction to Programming and Computational Problem Solving - 2

CSE 8B

CSE 8B: Introduction to Programming and Computational Problem Solving - 2

- Today
  - Course overview
  - Logistics

Introduction to programming courses

- A long time ago in a CSE department far, far away…
  - CSE 8A did not exist
  - CSE 8B did not exist
  - CSE 10: Introduction to Computer Science and Object-Oriented Programming using C++
  - CSE 11 did not exist
  - CSE 12: Basic Data Structures and Object-Oriented Design using C++

Introduction to programming courses

- Then, the **first** change: Java
  - CSE 8A did not exist
  - CSE 8B did not exist
  - CSE 10: Introduction to Computer Science and Object-Oriented Programming using C++
  - CSE 11: Introduction to Computer Science and Object-Oriented Programming using Java
  - CSE 12: Basic Data Structures and Object-Oriented Design using C++ and/or Java

Introduction to programming courses

- Then, the **second** change: no more C++
  - CSE 8A did not exist
  - CSE 8B did not exist
  - ~~CSE 10: Introduction to Computer Science and Object-Oriented Programming using C++~~
  - CSE 11: Introduction to Computer Science and Object-Oriented Programming using Java
  - CSE 12: Basic Data Structures and Object-Oriented Design using Java (and C++)

Introduction to programming courses

- Then, the **third** change: half-paced version of CSE 11 as two courses
  - CSE 8A: Introduction to Computer Science and Object-Oriented Programming using Java I
  - CSE 8B: Introduction to Computer Science and Object-Oriented Programming using Java II
  - CSE 11: Introduction to Computer Science and Object-Oriented Programming using Java
  - CSE 12: Basic Data Structures and Object-Oriented Design using Java (and C++)

Introduction to programming courses

- Last year, the **fourth** change: reboot CSE 8A, 8B, and 11
  - CSE 8A: Introduction to Programming and Computational Problem Solving - 1
  - CSE 8B: Introduction to Programming and Computational Problem Solving - 2 using Java
  - CSE 11: Introduction to Programming and Computational Problem Solving - Accelerated
  - CSE 12: Basic Data Structures and Object-Oriented Design using Java ~~(and C++)~~

Introduction to programming courses

- CSE 8A: Introduction to Programming and Computational Problem Solving - 1
  - Introduction to (procedural) programming
  - No specific programming language
- CSE 8B: Introduction to Programming and Computational Problem Solving - 2 using Java
  - Introduction to object-oriented programming
  - Does not assume knowledge of Java
- CSE 11: Introduction to Programming and Computational Problem Solving – Accelerated
  - CSE 8A + CSE 8B in one quarter

CSE 8B topics

- Introduction to Java
- Review fundamentals of programming
- Objects and classes
- Object-oriented thinking
- Inheritance
- Polymorphism
- Abstract classes
- Interfaces
- Class design guidelines
- Exception handling
- Assertions
- Text input/output (I/O)
- Binary I/O
- Recursion
- Event-driven programming

Introduction to Java and review fundamentals of programming

- Introduction to Java and programs
- Elementary programming
- Selections
- Mathematical functions, characters, and strings
- Loops
- Methods
- Single-dimension arrays and multidimensional arrays

Object-oriented programming

- Object-oriented programming (OOP) involves programming using objects
- **This is the focus of CSE 8B**

Objects and classes

- An object represents an entity in the real world that can be distinctly identified
  - For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects
  - An object has a unique identity, state, and behaviors
- Classes are constructs that define objects of the same type

Object-oriented thinking

- Classes provide more flexibility and modularity for building reusable software
- Class abstraction and encapsulation
  - Separate class implementation from the use of the class
  - The creator of the class provides a description of the class and let the user know how the class can be used
  - The user of the class does not need to know how the class is implemented
  - The detail of implementation is encapsulated and hidden from the user

Inheritance

- Suppose you will define classes to model circles, rectangles, and triangles
- These classes have many common features
- What is the best way to design these classes so to avoid redundancy?
- Object-oriented programming allows you to define new classes from existing classes
- This is called *inheritance*

Superclasses and subclasses

- Inheritance enables you to define a general class (i.e., a *superclass*) and later extend it to more specialized classes (i.e., *subclasses*)
- A subclass inherits from a superclass
  - For example, both a circle and a rectangle are geometric objects
    - `GeometricObject` is a superclass
    - `Circle` is a subclass of `GeometricObject`
    - `Rectangle` is a subclass of `GeometricObject`
- Models **is-a** relationships
  - For example
    - `Circle` **is-a** `GeometricObject`
    - `Rectangle` **is-a** `GeometricObject`

Polymorphism

- A class defines a type
- A type defined by a subclass is called a *subtype*, and a type defined by its superclass is called a *supertype*
  - For example
    - `Circle` is a subtype of `GeometricObject`, and `GeometricObject` is a supertype for `Circle`
- *Polymorphism* means that a variable of a supertype can refer to a subtype object
  - Greek word meaning “many forms”

Abstract classes

- Inheritance enables you to define a general class (i.e., a *superclass*) and later extend it to more specialized classes (i.e., *subclasses*)
- Sometimes, a superclass is so general it cannot be used to create objects
  - Such a class is called an *abstract class*
- An abstract class cannot be used to create objects
- An **abstract** class can contain abstract methods that are implemented in **concrete** subclasses
- Just like nonabstract classes, models **is-a** relationships
  - For example
    - `Circle` **is-a** `GeometricObject`
    - `Rectangle` **is-a** `GeometricObject`

Abstract classes and interfaces

- A superclass defines common behavior for **related** subclasses
- An *interface* can be used to define common behavior for classes, including **unrelated** classes
- Interfaces and abstract classes are closely related to each other

Interfaces

- An interface is a class-like construct that contains **only** constants and abstract methods
  - In many ways, an interface is similar to an abstract class, but the intent of an interface is to specify common behavior for objects
    - For example, you can specify that the objects are comparable and/or cloneable using appropriate interfaces
- Interfaces model **is-kind-of** relationships
  - For example
    - `Fruit` **is-kind-of** `Edible`

Exception handling

- Exceptions are errors caused by your program and external circumstances
  - These errors can be caught and handled by your program

Assertions

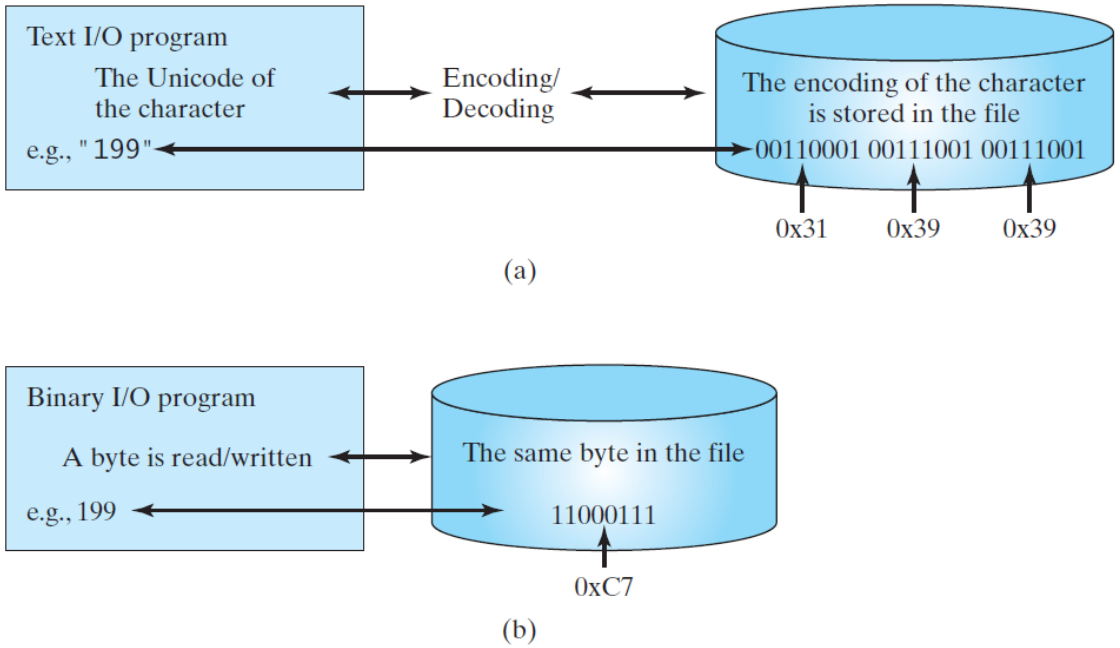
- An assertion is a Java statement that enables you to assert an assumption about your program
- An assertion contains a Boolean expression that should be true during program execution
- Assertions can be used to assure program correctness and avoid logic errors

Text I/O

- In order to perform I/O, you need to create objects using appropriate Java I/O classes
  - The objects contain the methods for reading/writing data from/to a file

Binary I/O

- Binary I/O does not involve encoding or decoding and thus is more efficient than text I/O



Recursion

- Recursion is a technique that leads to elegant solutions to problems that are difficult to program using simple loops
- A recursive method is one that invokes itself directly or indirectly

Event-driven programming

- An event is an object created from an event source
- You can write code to process events such as a button click, mouse movement, and keystrokes

Syllabus

- Instructor: Ben Ochoa
- TAs: Yundong Wang and Benson Budiman
- Tutors: Sumadhwa Guruprasad, Sophia Klueter, and Edward Wang
- Public course website
  - https://cseweb.ucsd.edu/classes/fa21/cse8B-a/
- Course is on Canvas
  - Piazza for discussion
  - Gradescope for submitting assignments
  - Media gallery for recorded lectures
- 19 lecture meetings
  - No lecture on Thanksgiving Eve (Wednesday, November 24)
- Weekly discussion section (optional)

Syllabus

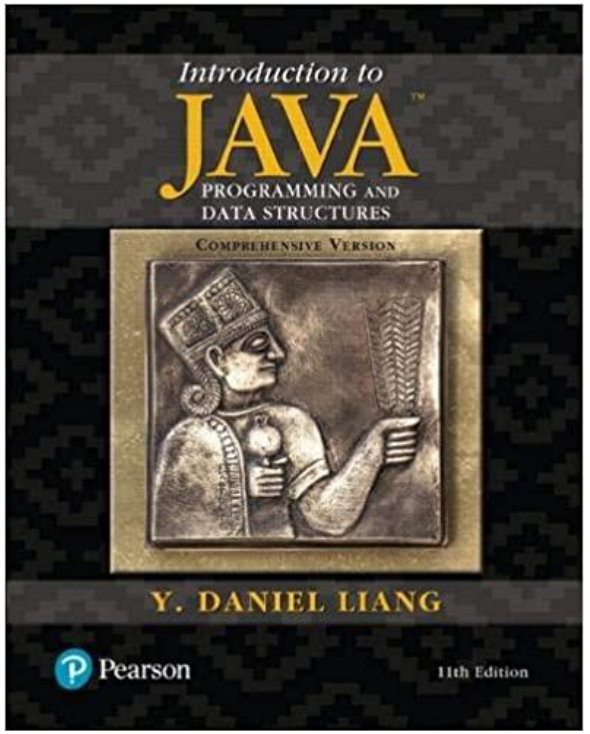
- Grading
  - 9 homework assignments (50% of grade)
    - Programming using Java
    - Late policy: 15% grade reduction for each 12 hours late
      - Will not be accepted 72 hours after the due date
    - Lowest assignment grade is dropped
  - 7 quizzes and final exam (50% of grade)
    - Quiz two days after assignments are due
      - 30 minutes, open notes/book, no collaboration
    - Final exam has 8 parts
      - First 7 parts correspond to quizzes and can only increase your grade, not decrease it (i.e., no fault)

Syllabus

- Grading
  - Extra credit
    - Education research study
      - Weekly survey (same day as quizzes)
      - 2% for participation, not your answers (so be honest)
    - Piazza
      - Ask (and answer) questions using Piazza, not email
      - Extensive, nontrivial participation could raise your grade (e.g., raise a B+ to an A-)

Textbook (optional)

- Introduction to Java Programming and Data Structures, 11th edition, comprehensive version
  - Y. Daniel Liang
- See book website
  - Errata



Collaboration Policy

It is expected that you complete your academic assignments on your (or your group's, if explicitly allowed for an assignment) own and in your (or your group's, if explicitly allowed for an assignment) own words and code. The assignments have been developed by the instructor to facilitate your learning and to provide a method for fairly evaluating your knowledge and abilities (not the knowledge and abilities of others). So, to facilitate learning, you are authorized to discuss assignments with others; however, to ensure fair evaluations, you are not authorized to use the answers developed by another, copy the work completed by others in the past or present, or write your academic assignments in collaboration with another person (or group, if explicitly allowed for an assignment). On quizzes or exams, collaboration or copying of any kind is not allowed.

Academic Integrity Policy

Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and in so doing protect the validity of University intellectual work. For students, this means that all academic work will be done by the individual (or group, if explicitly allowed for an assignment) to whom it is assigned, without unauthorized aid of any kind.

Academic Integrity Policy

You should not attempt to search for homework solutions online or in sources outside of the course text. If you accidentally stumble upon a homework solution in an outside source you must cite it in your homework solution. If your solution proves to be too similar to the cited one, you may lose credit on the problem; however, failure to cite the other solution will be treated as an academic integrity violation.

Academic Integrity Violation

If the work you submit is determined to be other than your own (or your group's, if explicitly allowed for an assignment), you will be reported to the Academic Integrity Office for violating UCSD's Policy on Integrity of Scholarship. In accordance with the CSE department academic integrity guidelines, ***students found committing an academic integrity violation on a homework assignment will receive a 0 on the assignment. Students found committing an academic integrity violation on a quiz or exam will receive an F in the course.***

Wait list

- Number of enrolled students is limited by
  - Number of instructional assistants (TAs and tutors)
- General advice
  - Wait for as long as you can
- UCSD policy: concurrent enrollment (Extension) students have lowest priority

Java Software Development

Java Development Kit (JDK)

- Develop on your own computer or UCSD Linux Cloud
  - https://linuxcloud.ucsd.edu
- For CSE 8B, we will be using Java 8. You must use one of the following JDKs
  - Oracle Java SE Development Kit 8 Update 301 (i.e., 8u301)
  - Open Java Development Kit (OpenJDK) 8u302
- OpenJDK 8u302 is already installed and configured on your UCSD Linux Cloud class account
- If using your own computer, then download and install one of the above JDKs
- **Your source code must compile and run from the command-line on your UCSD Linux Cloud class account**

Text Editor

- Use whatever text editor you want
  - Notepad, Notepad++, vi (or Vim), Emacs, etc.
- You can also use a more advanced source code editor (e.g., Visual Studio Code), but the instructional team will not assist you in configuring it
- **Your source code must compile and run from the command-line on your UCSD Linux Cloud class account**
  - Beware of integrated development environments (IDEs) that generate source code!

Next Lecture

- Introduction to Java and programs
- Elementary programming
- Reading:
  - Liang
    - Chapters 1 and 2