Introduction and Overview

Introduction to Programming and Computational Problem Solving II

CSE 8B

Lecture 1

CSE 8B: Introduction to Programming and Computational Problem Solving II

- Today
  - Course overview
  - Logistics

Introduction to programming courses

- CSE 8A: Introduction to Programming and Computational Problem Solving I
  - Introduction to **procedural programming**
  - Switched from Java to Python a few years ago
- CSE 8B: Introduction to Programming and Computational Problem Solving II
  - Introduction to **object-oriented programming**
  - Uses Java
- CSE 11: Introduction to Programming and Computational Problem Solving: Accelerated Pace
  - CSE 8A topics + CSE 8B in one quarter
  - Uses Java

CSE 8B topics

- Introduction to Java
- Numbers and mathematics
- Characters and strings
- Selections
- Methods
- Loops
- Recursion (simple)
- Arrays

Procedural programming

Object-oriented programming

- Objects and classes
  - Object-oriented thinking
- Inheritance
- Polymorphism
- Abstract classes
- Interfaces
- Introduction to generics
- Exceptions
- Text file input/output
- Binary file input/output
- Assertions

Introduction to Java

- Java is:
  - a high-level programming language
    - Computer-specific details are abstracted
  - an object-oriented programming language
    - Based on classes
  - a strongly typed language
    - **Programmers must explicitly identify the type of every variable, method, and object**
  - a general-purpose programming language
    - Not specialized to a particular application domain
  - platform independent
    - Write a program once and run it on any computer

Procedural programming

- Procedural programming involves programming the steps a program must take to reach a desired state
- This is the focus of CSE 8A

Numbers and mathematics

- Numerical data types (e.g., an integer)
- Numeric operations (e.g., addition)
- Mathematical functions (e.g., cosine)
- Reading numbers from the console

Characters and strings

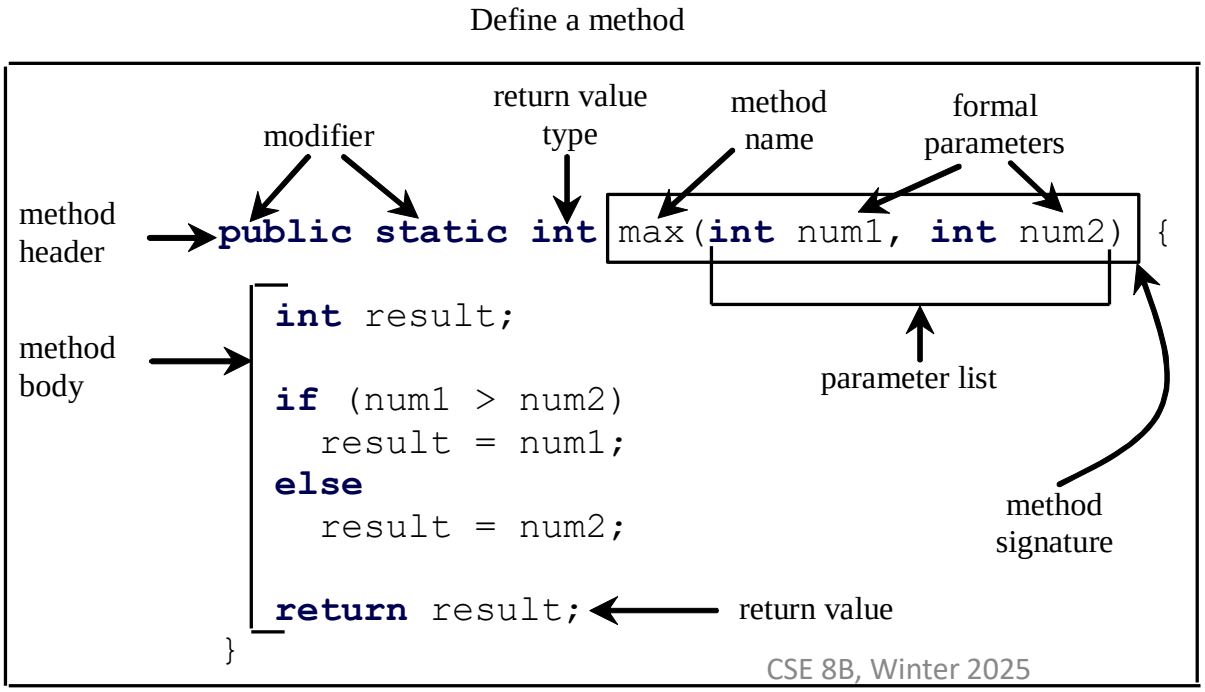
- Character data type (i.e., `char`)
- Comparing and testing characters
- String data type (i.e., `String`)
- Simple string methods (e.g., number of characters in a string)
- Reading a character and string from the console

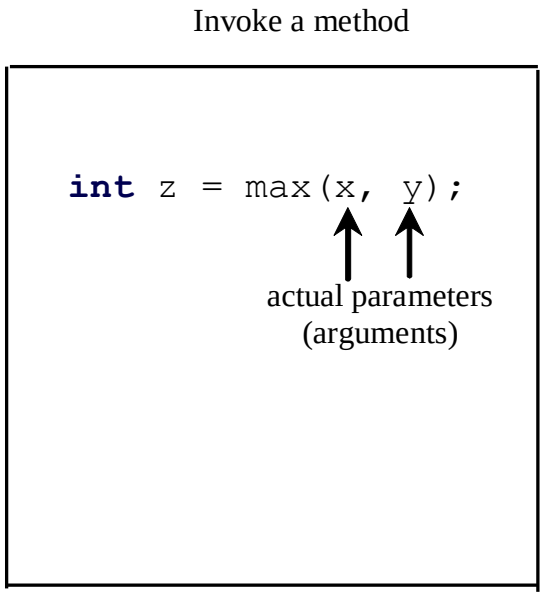
Selections

- Relational operators (e.g., less than, equal to)
- Logical operators (e.g., not, and, or)
- `if` statements
- `if-else` statements
- `switch` statements

Methods

- A method is a collection of statements that are grouped together to perform an operation
- Write a method once and reuse it anywhere



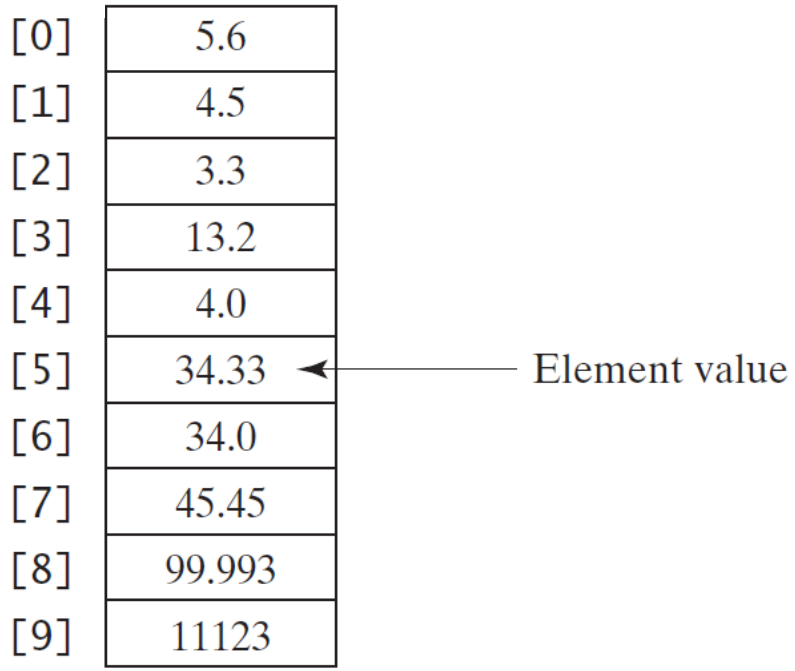


Loops and recursion

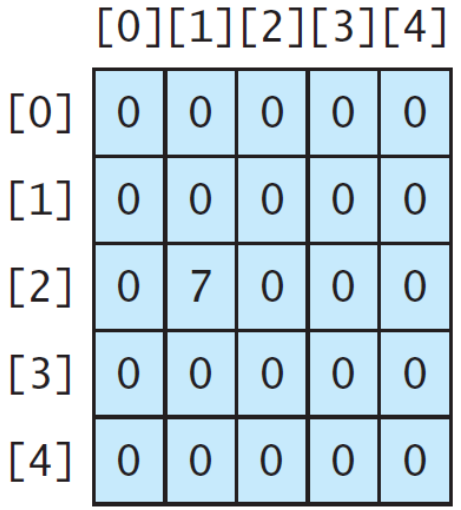
- `while` loops
- `do-while` loops
- `for` loops
- Recursion is a technique that leads to elegant solutions to problems that are difficult to program using simple loops
  - A recursive method is one that invokes itself directly or indirectly

Arrays

- Array is a data structure that represents a collection of the same types of data



1-dimensional array



2-dimensional array

Object-oriented programming

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

Procedural programming vs object-oriented programming

- Procedural programming
  - Data and operations on data are separate
  - Requires passing data to methods
- Object-oriented programming
  - Data and operations on data are in an object
  - Organizes programs like the real world
    - All objects are associated with both attributes and activities
  - Using objects improves software reusability and makes programs easier to both develop and maintain

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 define classes to model circles, rectangles, and triangles
- These classes have many common features
- What is the best way to design these classes 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`
    - `Fish` **is-kind-of** `Edible`

Additional topics

- Introduction to generics
- Exceptions
- Text file input/output (I/O)
- Binary file input/output (I/O)
- Assertions

Introduction to generics

- Generics let you parameterize types
  - You can define a method or class with generic types, which are replaced with concrete types
- **CSE 8B and 11 only use Java built-in generics, we will not be defining our own generics**

Exceptions

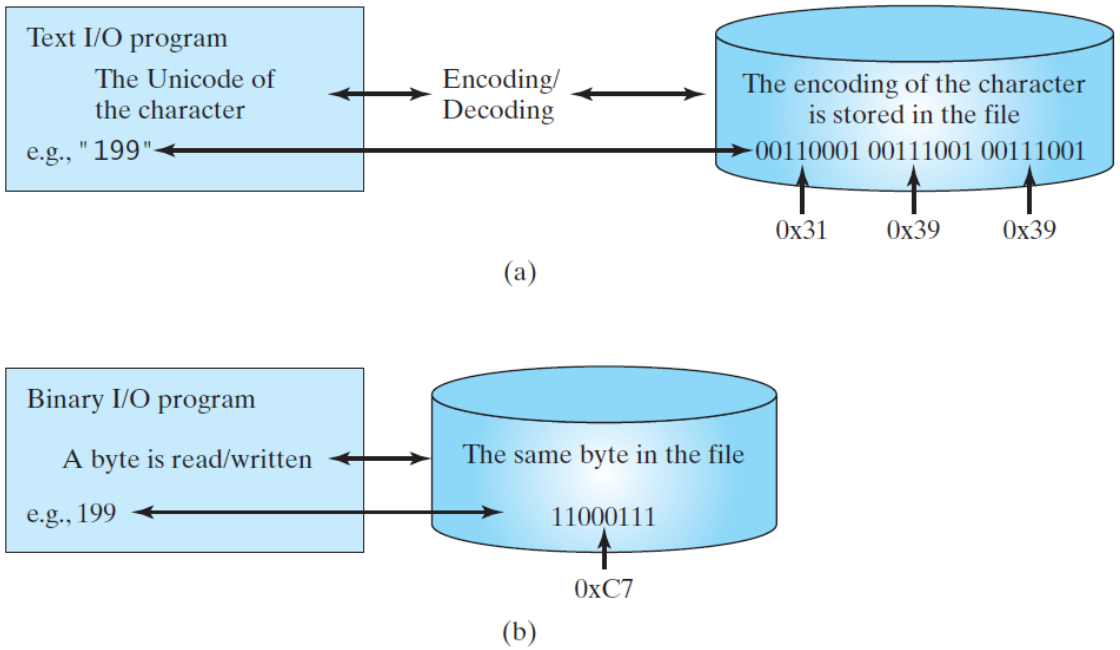
- Exceptions are errors caused by your program and external circumstances
  - These errors can be caught and handled by your program

Text file input/output (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 file input/output (I/O)

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



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

Syllabus

- Instructor: Ben Ochoa
- TAs: Sumadhwa (Sumu) Guruprasad, Rachel Lim, and Hailey Li
- Tutors: Hunter Flores, Jordan Ruggles, Neelkanth Shitolay, Jacob Takesuye, Cedric-James David, Sonia Fereidooni, Jerry Gong, Kile Hsu, April Huang, Brandon Jonathan, Pranav Kambhampati, Richard Nie, Trey Shneour, Terri Tai, Jeffrey Thi, Luis Xu, and Karen Yan
- Public course website
  - https://cseweb.ucsd.edu/classes/wi25/cse8B-a/
- Course is on Canvas
  - **Quizzes** for prelecture quizzes
  - **Piazza** for discussion
  - **Gradescope** for submitting assignments
  - **Media gallery** for prelecture videos, and recorded lecture and discussion meetings
- 17 lecture meetings
  - No lecture meeting on Jan 20 (Martin Luther King, Jr., Holiday)
  - No lecture meeting on Feb 17 (Presidents' Day Holiday)
  - Optional lecture meeting on Mar 10
- Weekly discussion section meeting

Syllabus

- Grading
  - Prelecture quizzes (10% of grade; 0.625% per 16 lectures)
    - Open from sometime two days before lecture until start of lecture
  - 7 homework assignments (45% of grade; can be upgraded)
    - 3.462% for assignment 1; 6.923% for assignments 2-7
    - Late policy: 15% grade reduction for each 12 hours late
      - Will not be accepted 48 hours after the due date
    - In order to pass the class, you must receive a passing grade for your overall assignments
  - Midterm assessment (20% of grade; can be upgraded with final assessment part 1)
    - Procedural programming (lectures 2-8, assignments 1-4)
  - Final assessment (25% of grade)

Syllabus

- Grading
  - Upgrades
    - Assignments
      - Except for assignment 1, after an assignment has been graded, there will be a resubmission period enabling you to earn up to 50% of the remaining points. If your points on the resubmitted assignment is **greater** than your points on the original assignment, then your assignment points will be replaced by the average of your original and resubmitted points (i.e., your points will **increase**); otherwise, your original assignment points will not change (i.e., **no fault**).
    - Midterm assessment
      - The final assessment will consist of 2 parts. Part 1 of the final assessment covers the same course material as the midterm assessment. If your grade on part 1 of the final assessment is **greater** than your grade on the midterm assessment, then your midterm assessment grade will be replaced by the average of your final assessment part 1 grade and your original midterm assessment grade (i.e., your midterm assessment grade will **increase**); otherwise, your midterm assessment grade will not change (i.e., **no fault**).

Syllabus

- The practice test, midterm assessment, and final assessments will take place in the Triton Testing Center's (TTC's) Computer-Based Testing Facility (CBTF; AP&M B349). The assessments will be administered by the TTC and follow the TTC's CBTF testing policies and procedures. **You must schedule a time to take your assessments in advance, and it is recommended you schedule them as soon as possible.** To schedule, visit PrairieTest and log in with your UC San Diego credentials.
- Students requesting accommodations for this course due to a disability must provide a current Authorization for Accommodation (AFA) letter (paper or electronic) issued by the Office for Students with Disabilities (OSD). Students are required to discuss accommodation arrangements with instructors and OSD liaisons in the department **in advance** of any assignments or assessments. Students with approved accommodations will be taking their exams at the TTC (PCYNH 364), where exams must be scheduled at least 72 hours in advance.

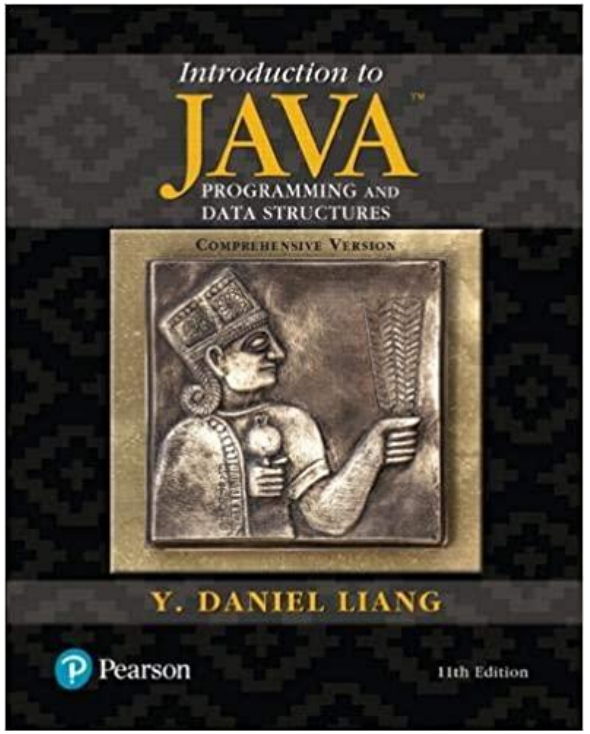
Syllabus

- Grading
  - Extra credit
    - If any opportunities arise, then there will be no more than 4% added to overall class grade
    - 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)

Lecture slides are a comprehensive summary of relevant book material

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



Collaboration policy

- Ask and answer questions on Piazza, not email
- Post **publicly** (optionally anonymously)
  - Conceptual questions and high-level questions about assignments
- All other posts must be **private** to "Instructors" (includes instructor and instructional assistants)
  - Low-level, detailed assignment questions (e.g., implementation details)
  - Assignment-specific code
  - Results (intermediate or final; e.g., numerical values, images, figures)
  - **Posting such items publicly is an academic integrity violation**
- If you are unsure, then post privately to "Instructors"
  - If suitable, then it will be changed to a public post
- Piazza is the official, University-sanctioned discussion forum
  - **Do not use Piazza to solicit others to an alternative forum**

Collaboration policy

It is expected that you complete your academic assignments on your own and in your own words and code. The assignments have been developed by the instructional team 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. On midterm and final assessments, 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 to whom it is assigned, without unauthorized aid of any kind. No student shall allow any academic work or academic credit to be completed or obtained, in part or in whole, for themselves by another (human or machine/artificial intelligence).

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, you will be reported to the Academic Integrity Office for violating UC San Diego'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 midterm or final assessment will receive an F in the course.***

Student conduct policy

UC San Diego strives to maintain a climate of fairness, cooperation, and professionalism. It is expected that you practice basic principles, including, but not limited to, mutual respect, civility, and decency, towards maintaining an atmosphere free of abusive or demeaning treatment. Non-academic student misconduct will be reported to the Center for Student Accountability, Growth, and Education for violating UC San Diego's Principles of Community.

Wait list

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

Certification of commencement of academic activity

- Every course at UC San Diego, per the US Department of Education, is required to certify whether students have commenced academic activity for a class to be counted towards eligibility for Title IV federal financial aid. This certification must be completed during the first two weeks of instruction.
- For CSE 8B, this requirement will be fulfilled via an ungraded prior knowledge quiz, which will assist the instructional team by providing information about your background coming into the course
  - In Canvas (https://canvas.ucsd.edu), go to the CSE 8B course and navigate to Quizzes, then click on **First Day Survey: Prior Knowledge #FinAid**

Next lecture

- Introduction to Java