

Introduction and Overview

Introduction to Programming and
Computational Problem Solving - 2

CSE 8B

Lecture 1

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

- Today
 - Course overview
 - Logistics
- Educational research study (extra credit)
 - Pre-test today, last 20 minutes of lecture meeting

Introduction to programming courses

- CSE 8A: Introduction to Programming and Computational Problem Solving - 1
 - Introduction to **procedural programming**
 - Switched from **Java** to **Python** a couple years ago
- CSE 8B: Introduction to Programming and Computational Problem Solving - 2
 - Introduction to **object-oriented programming**
 - Uses **Java**
- CSE 11: Introduction to Programming and Computational Problem Solving – Accelerated
 - CSE 8A + CSE 8B in one quarter
 - Uses **Java**

CSE 8B topics

Object-oriented programming

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

Procedural 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

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

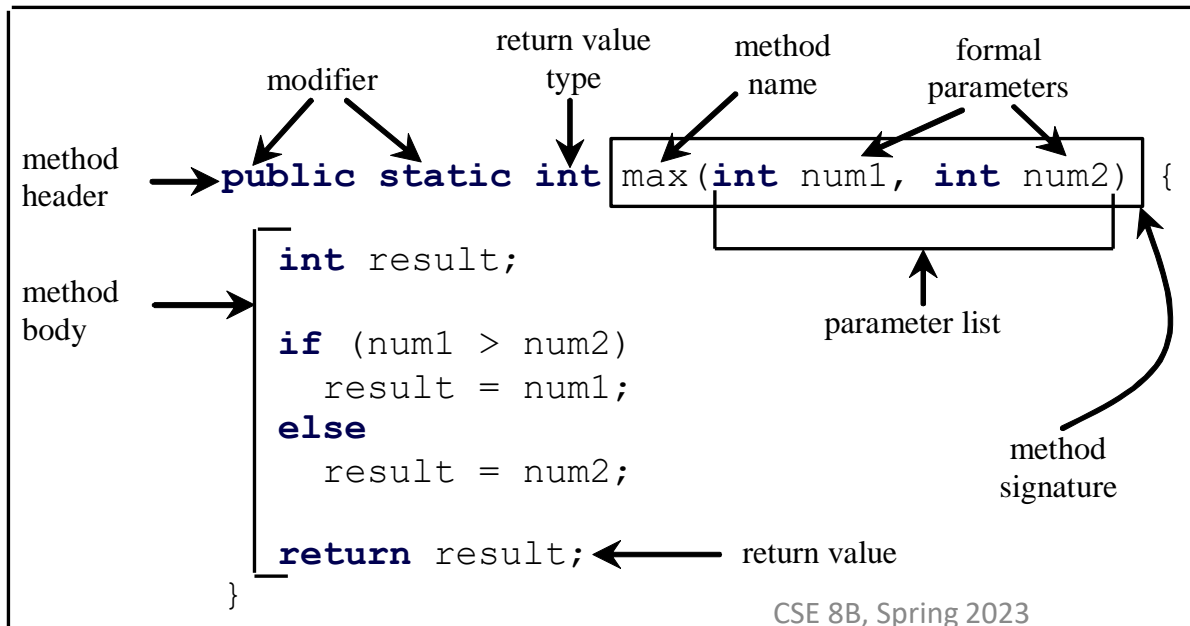
Selections

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

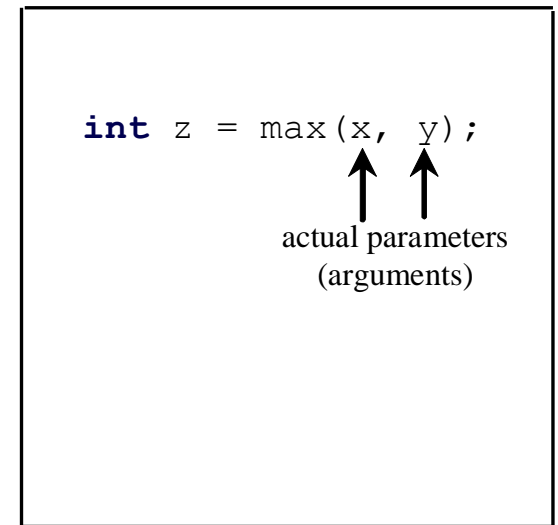
Methods

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

Define a method



Invoke a method



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

| | |
|-----|--------|
| [0] | 5.6 |
| [1] | 4.5 |
| [2] | 3.3 |
| [3] | 13.2 |
| [4] | 4.0 |
| [5] | 34.33 |
| [6] | 34.0 |
| [7] | 45.45 |
| [8] | 99.993 |
| [9] | 11123 |

← Element value

1-dimensional array

| | [0] | [1] | [2] | [3] | [4] |
|-----|-----|-----|-----|-----|-----|
| [0] | 0 | 0 | 0 | 0 | 0 |
| [1] | 0 | 0 | 0 | 0 | 0 |
| [2] | 0 | 7 | 0 | 0 | 0 |
| [3] | 0 | 0 | 0 | 0 | 0 |
| [4] | 0 | 0 | 0 | 0 | 0 |

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
- **In CSE 8B, we will only be using Java built-in generics, not defining our own generics**

Exceptions

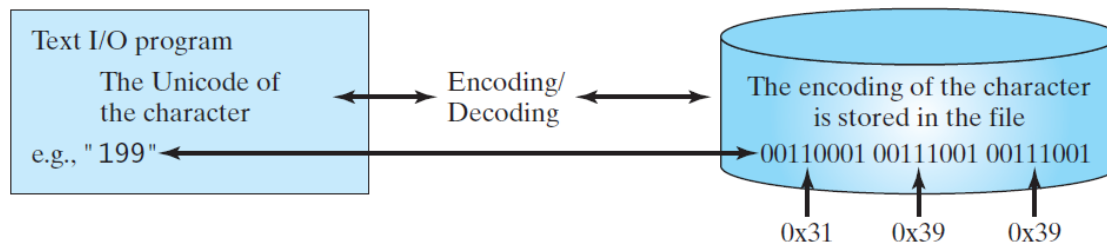
- Exceptions are errors caused by your program and external circumstances
 - These errors can be caught and handled by your program

Text file input/output (I/O)

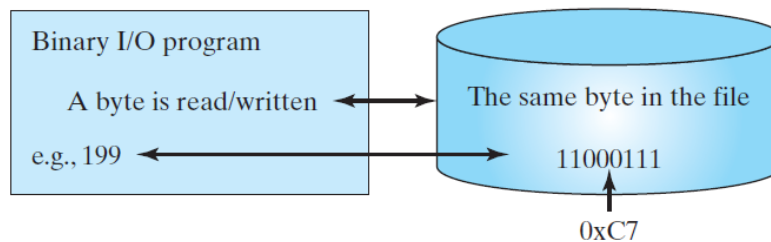
- In order to perform I/O, you need to create objects using appropriate Java I/O classes
 - The objects contain the methods for reading/writing data from/to a file

Binary file input/output (I/O)

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



(a)



(b)

Assertions

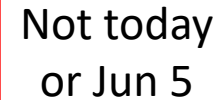
- An assertion is a Java statement that enables you to assert an assumption about your program
- An assertion contains a Boolean expression that should be true during program execution
- Assertions can be used to assure program correctness and avoid logic errors

Syllabus

- Instructor: Ben Ochoa
- TAs: Darren Yeung and Brian Nguyen
- Tutors: Edward Wang, Sumadhwa Guruprasad, and Mingyi Li
- Public course website
 - <https://cseweb.ucsd.edu/classes/sp23/cse8B-a/>
- Course is on Canvas
 - **Quizzes** for prelecture quizzes
 - **Piazza** for discussion
 - **Gradescope** for submitting assignments
 - **Media gallery** for recorded lectures
- 19 lecture meetings
 - No lecture meeting on May 29 (Memorial Day observance)
- Weekly discussion section (optional)

Syllabus

Not today
or Jun 5



- Grading
 - Lecture participation (8% of grade; 0.5% per 16 lectures)
 - Prelecture quizzes (12% of grade; 0.75% per 16 lectures)
 - Open from day before lecture until start of lecture
 - 8 homework assignments (40% of grade; can be upgraded)
 - 2% for assignment 1; 4.75% for assignments 2-8
 - Late policy: 15% grade reduction for each 12 hours late
 - Will not be accepted 48 hours after the due date
 - Midterm exam (20% of grade; can be upgraded with final exam part 1)
 - Procedural programming (lectures 2-8, assignments 1-4)
 - Final exam part 2 (20% of grade)
 - Object-oriented programming (lectures 9-18, assignments 5-8)

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 autograder points (i.e., not the manually graded points; they cannot be upgraded). If your autograder points on the resubmitted assignment is **greater** than your autograder points on the original assignment, then your assignment autograder points will be replaced by the average of your original and resubmitted autograder points (i.e., your autograder points will **increase**); otherwise, your original assignment points will not change (i.e., **no fault**).
 - Midterm exam
 - If your grade on part 1 of the final exam is **greater** than your grade on the midterm exam, then your midterm exam grade will be replaced by the average of your final exam part 1 grade and your previous midterm exam grade (i.e., your midterm grade will **increase**); otherwise, your midterm exam grade will not change (i.e., **no fault**).

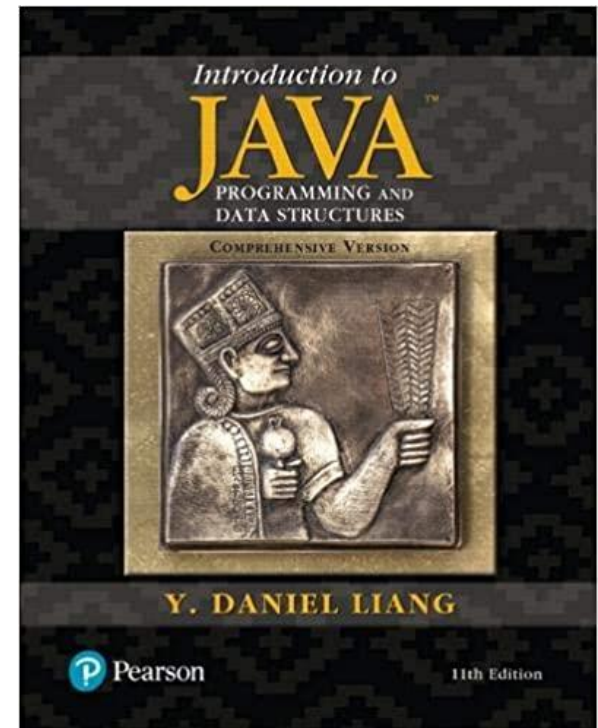
Syllabus

- Grading
 - Extra credit
 - Education research studies
 - Pre-test (**today**) and post-test (near end of quarter)
 - Weekly survey (on Fridays)
 - 3% for participation, not your answers
 - 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

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 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 an exam will receive an F in the course.***

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
- UCSD 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