University of Pittsburgh

Algorithms and Data Structures 1
CS 0445

Fall 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

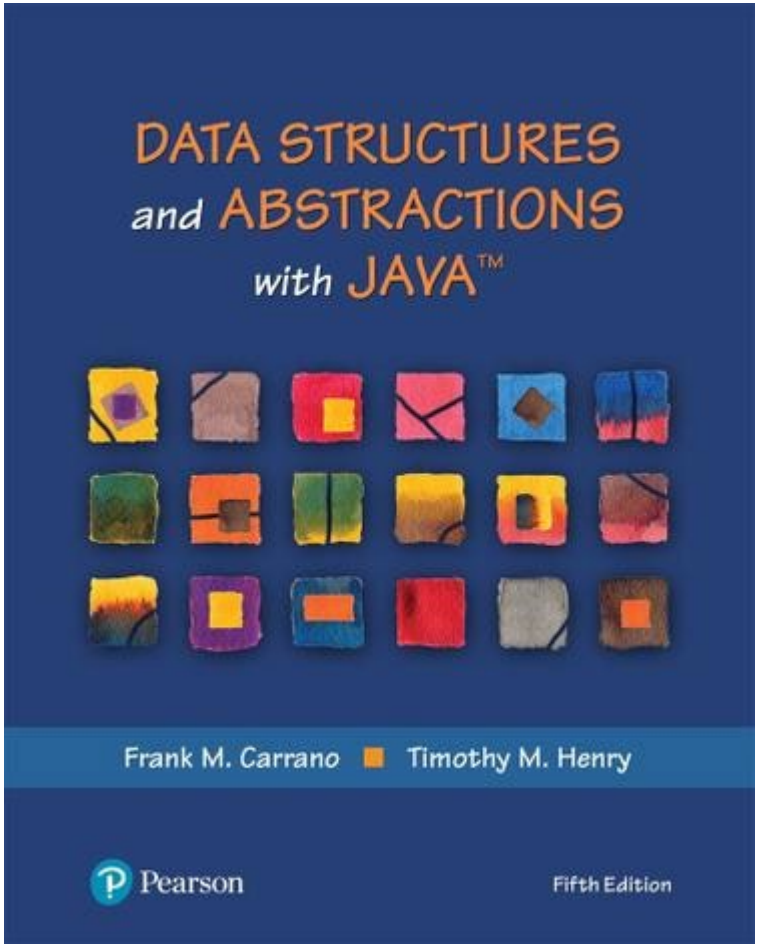
Contact Info

- **Course website:** http://www.cs.pitt.edu/~skhattab/cs0445/
- **Instructor:** Sherif Khattab ksm73@pitt.edu
- **My Student Support Hours:** https://khattab.youcanbook.me
  - MW: 10:00-12:00; TuTh: 13:00-15:00; F by appointment
  - 6307 Sennott Square, Virtual Office: https://pitt.zoom.us/my/khattab
  - Please schedule at: https://khattab.youcanbook.me/
- **Teaching Team:**
  - Radley Lettich, ral109@pitt.edu
  - Julia Malnak, jum97@pitt.edu
  - Evan Kozierok, eak80@pitt.edu
  - More TAs to come
- No recitations this week, but you got some work to do!
- **Communication**

Piazza (**Please expect a response within 72 hours)**

Email not recommended!

Textbook



Data Structures and Abstractions with Java (5th Edition)

Frank M. Carrano and Timothy M. Henry

Grades

- 40% on best four out of five programming assignments; mostly autograded
  - posted on Canvas, distributed using Github, and submitted on **Gradescope** from Github
- 20% on homework assignments on Gradescope
- 20% on exams: 12% on higher grade and 8% on lower
- 10% on lab exercises; mostly autograded
- 10% on in-class Top Hat questions

Canvas Walkthrough

- Lectures posted on Tophat
  - Draft slides available on Github
- Lecture and recitation recordings
  - under Panopto Video
- RedShelf Inclusive Access for the Textbook
  - You can cancel before Add/Drop
- Piazza for discussion and communication
- Gradescope and autograding policies
- Academic Integrity
- NameCoach

Expectations

- Your continuous feedback is important!
  - Anonymous Qualtrics survey
  - Midterm and Final OMET
- Your engagement is valued and expected with
  - classmates
  - teaching team
  - material

Lecture structure (mostly)

Time	Description
~5 min before and after class	Informal chat
~25 min	Announcements, review of muddiest points on previous lecture, and QA on assignments/labs/homework problems
~45 min	Lecturing with Tophat questions and/or activities
~5 minutes	QA and muddiest points/reflections

How to success in this course

- Attend lectures and recitations (if you absolutely cannot attend, watch the video recordings)
- Study often!
- Put effort into the weekly homework assignments
- Refresh your Java programming (CS 0401) and **debugging skills**
- Start early and show up to student support hours!

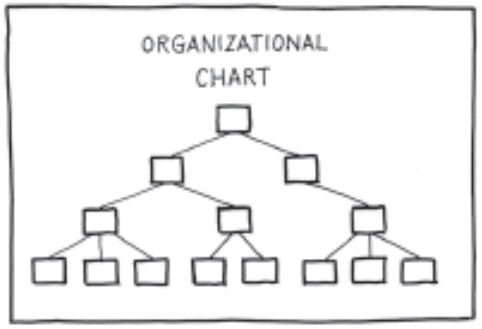
Announcements

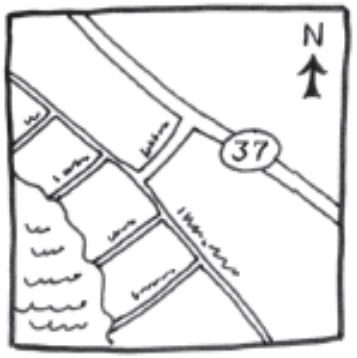
- Lab 0 is due this Friday (not graded)
- Recitations start next week
- Homework 1 will be assigned this Friday
- JDB Example will be available on Canvas
- Draft slides and handouts available on Canvas

Today's Agenda

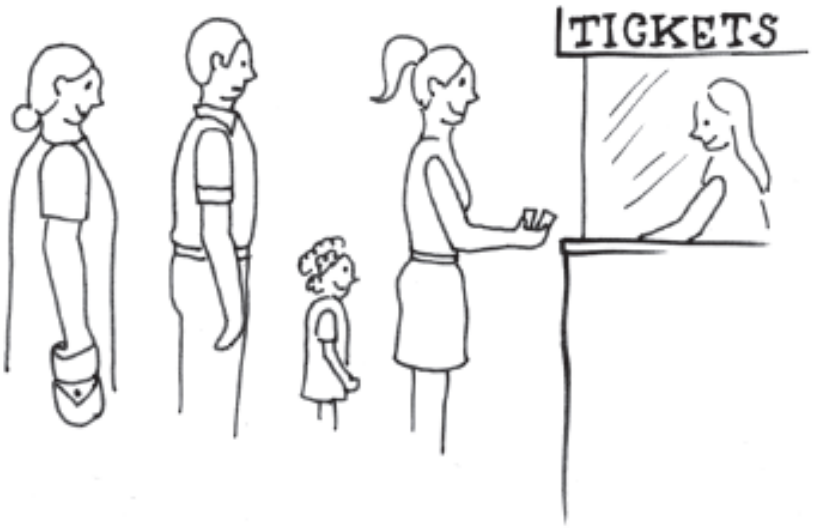
- Course goals and overview
- Java Review
  - Encapsulation and Abstraction
  - Reference types
  - Class Design
    - Composition
      - Clone
    - Inheritance
      - Polymorphism
  - Abstract Data Types
    - Java Interfaces
    - Generics
  - File Operations

Data Organization in Life





Examples of everday data organizations



Data Organization in Computers

- In many cases, data are organized in computers as a Collection of data items with operations on them
  - Bag
  - List
  - Stack
  - Queue
  - Dictionary
  - Tree
  - Graph
- All these are examples of Abstract Data Types (ADTs)
- Implemented by one or more Data Structures

Code in this Course

- Client code
  - Code that uses ADTs
- Library code
  - Code that implements ADTs
  - Use Java features to help us
    - Java Interfaces
    - Java Generics

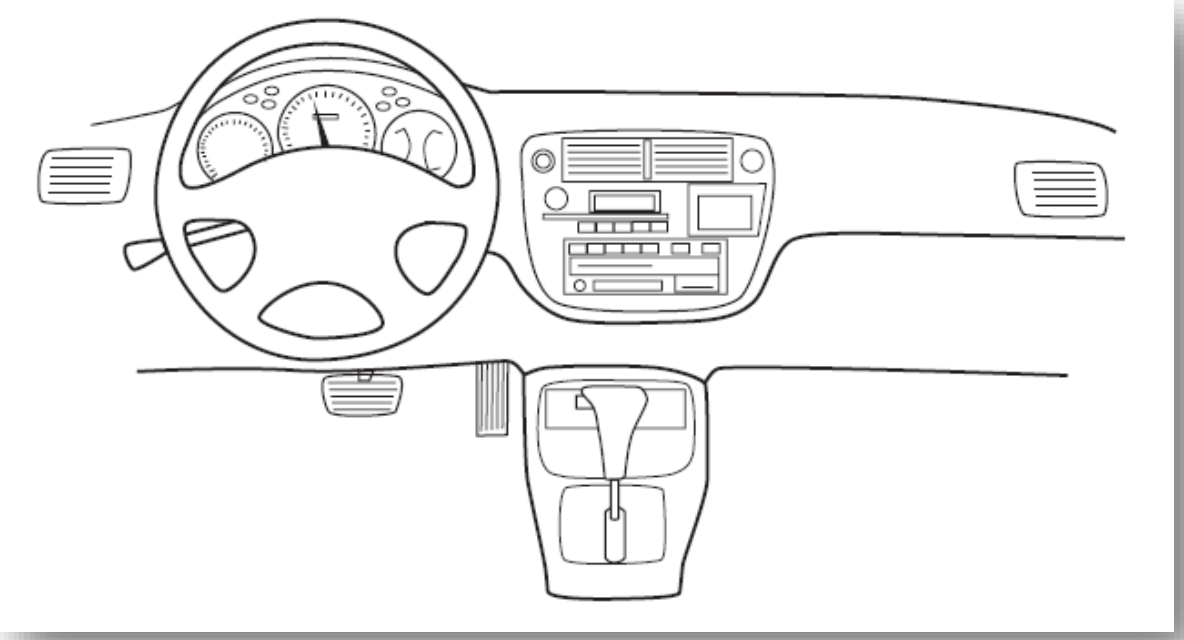
Course Goals

- Implement algorithms and applications that use Data Structures (Client-side)
  - e.g., Sorting, Searching
- Implement Fundamental Data Structures (Library Developer-side)
  - Bag, List, Stack, Hash Table, Queue
- Use Recursion for problem solving
- Analyze the running-time of
  - operations on Data Structures
  - algorithms that use Data Structures

Encapsulation

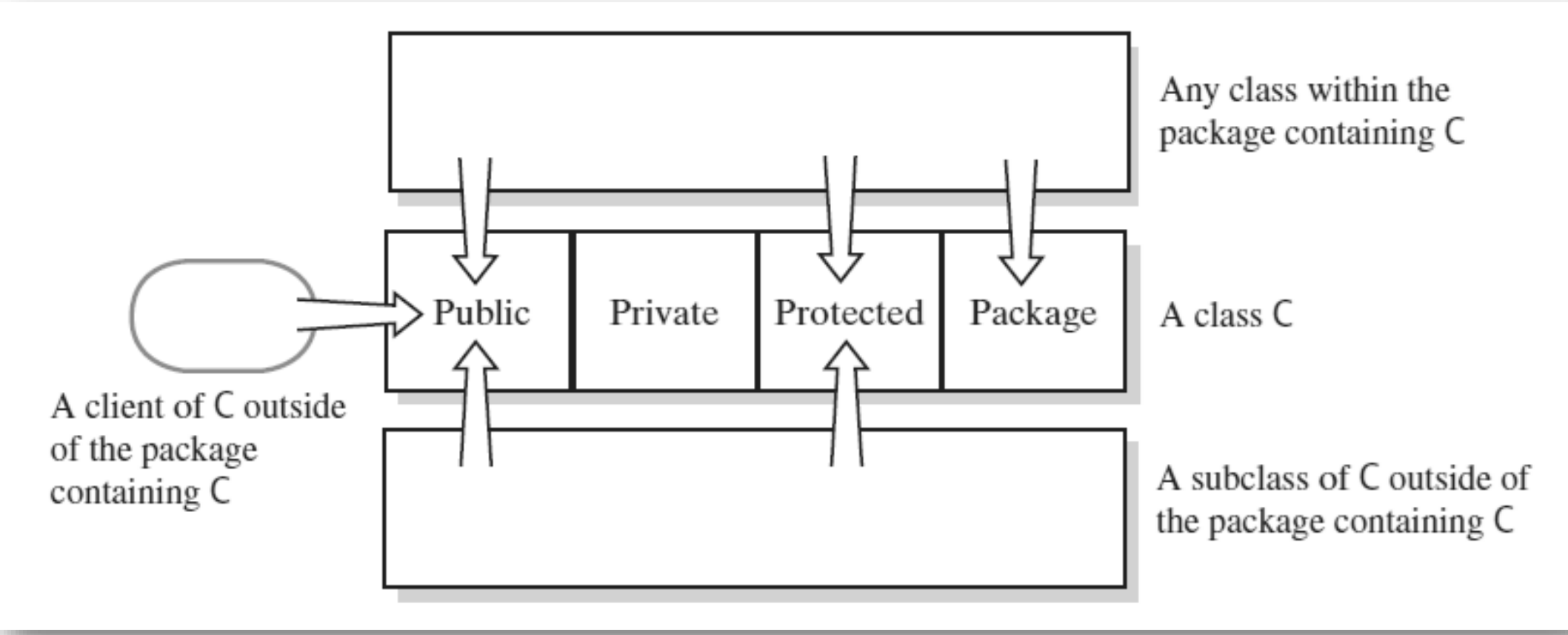
- Enclose data and methods within a class
- Hide implementation details
- Programmer receives only enough information to be able to use the class

An automobile's controls are visible to the driver, but its inner workings are hidden



Access Modifiers

- Public, private, protected, and package access of the data fields and methods of class `C`

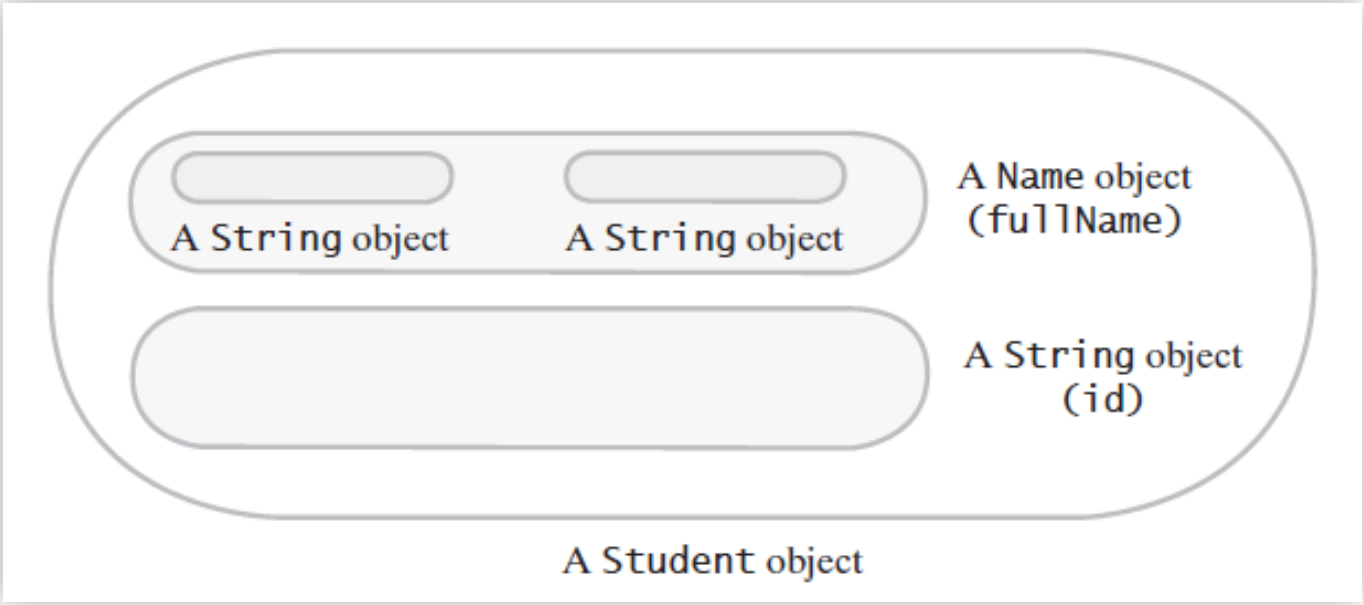


Composition

- A class uses composition when it has a data field that is an instance of another class
- Composition is a “has a” relationship
- Consider a class of students, each has
  - A name, an identification number.
- Thus, class Student contains two objects as data fields:
  - An instance of the class Name
  - An instance of the class String

Composition

- A Student object is composed of other objects



clone

- A Method of the Class **Object**
- Takes no arguments and returns a copy of the receiving object

Composition Code Example

Please Check Square.java and Main.java under Live Code\Introduction\Take1 folder.

Inheritance Code Example

Please Check Square.java, ColoredSquare.java, and Main.java under Live Code\Introduction\Take2 folder.

Invoking Constructors from Within Constructors

- Constructors typically initialize a class's data fields
- To call constructor of superclass explicitly:
  - Use **super( )** within definition of a constructor of a subclass
- If you omit **super( )**
  - Constructor of subclass automatically calls default constructor of superclass.

Overriding and Overloading Methods

- When a subclass defines a method with
  - the same name
  - the same number and types of parameters
  - and the same return type as a method in the superclass

    …
- Then definition in the subclass is said to *override* the definition in the superclass.
- You can use **super** in a subclass to call an overridden method of the superclass.

Overriding and Overloading Methods

- Possible to have new method invoke the inherited method
  - Need to distinguish between the method for subclass and method from superclass

```
public String toString()
{
   return super.toString() + ", " + degree + ", " + year;
} // end toString
```