The University of Texas at El Paso
Department of Computer Science
CS3331 – Object-Oriented Programming (In-Class)
Starbucks Sales Tracker
Fall 2025

Learning Objectives

By the end of this activity, students should:

- Load and parse CSV data into an internal model.
- Apply **abstraction** to express shared drink behavior once.
- Use **inheritance** to represent specific drink categories.
- Enforce **encapsulation** to protect internal state and validate input.
- Leverage **polymorphism** so pricing/labels/behavior come from the category, not conditionals.
- Use **composition** to build orders from items and summarize sales.

Generative AI: Students are permitted to use Generative AI tools for this assignment. They are not permitted to simply copy/paste code from Gen AI. Students should use Generative AI as a tool to help them with parts of the assignment that they need help with, but most of the code should be written by the student. Use Gen AI as a helper for syntax, general implementation ideas, but not to produce the entire assignment.

Students are permitted to work in pairs (2 people)

Prior to coding anything, draw out (rough sketch) a UML Class Diagram of the system.

Background & Scenario (unchanged content)

Starbucks wants to keep track of drink sales in a simple digital system. They have provided a CSV file that contains menu information:

- Drink name (e.g., *Caffè Latte*)
- Drink type (e.g., *coffee, tea, refresher, frappuccino*)
- Drink size (e.g., *Tall, Grande, Venti*)
- Price

Your job is to write a simple **command line interface (CLI)** program to simulate this sales tracker.

Constraints

- **Abstraction:** Define a general drink concept with common attributes (e.g., name, size, base price) and overridable operations (e.g., computing price).
- **Inheritance:** Create several specific drink categories that extend the general concept. Categories may customize behavior (e.g., pricing rules).
- **Encapsulation:** Keep drink/order data private/internal; expose only what is necessary.
- **Polymorphism:** The CLI should treat all drinks through the general concept; behavior differences come from the category's override, not from if/else chains on type strings.
- **Composition**

Keep naming and structure your choice—just meet the requirements above.

Part 1 — Setup

1. Read menu.csv once at program start.
2. Convert each row into a drink instance belonging to the appropriate category.
3. Store numeric prices as numbers.

Part 2 — CLI Menu (loop until quit)

Provide options functionally equivalent to:

1. Show all available drinks.
2. Search for drinks by type (e.g., coffee, tea).
3. Place an order (one or more drinks).
4. View today's sales summary.
5. Quit.

Part 3 — Functionality Requirements

Show all available drinks

- Print in a readable table. Use each drink's own label/price.
- Avoid duplicates.

Search by type

- Prompt for a type and display matching drinks.

Place an order

- Accept repeated inputs such as name,size,quantity until the user indicates checkout.
- Validate that the requested drink exists (by name and size). Handle errors gracefully.
- Add each requested item to the order.
- Compute a running total using the drink's own pricing behavior.

View today's sales summary

- Total number of drinks sold.
- Total revenue.
- Most popular drink (by name+size).
- Unique drink types sold today.
- Drinks present in the menu but not sold today.

Part 4 — Additional Features (Optional if Time Allows)

- Bulk discount: Reduce order total by a percentage (e.g. 10%) when the order contains more than 3 total drinks.
- Happy Hour for a category: Reduce price for one category (e.g., teas) when a flag is active.

Express discounts either by category-specific pricing behavior or by a single pluggable calculation step in the order. Do **not** hard-code large conditional chains.

Implementation Guidance

- Parse CSV into the menu collection at startup.
- When computing totals or displaying labels, invoke behavior on the drink itself, rather than switching on type strings.
- Maintain a session record of completed orders to drive the summary.

Error Handling & Edge Cases

- Unknown drink, invalid size, or non-numeric quantity -> show a helpful message and reprompt.
- Empty order at checkout -> warn and allow the user to keep adding or cancel.
- CSV lines with missing fields -> skip with a warning (do not crash).