REPRESENTING DATA IN YOUR PROGRAM

CSC 207 SOFTWARE DESIGN

Computer Science
UNIVERSITY OF TORONTO

LEARNING OUTCOMES

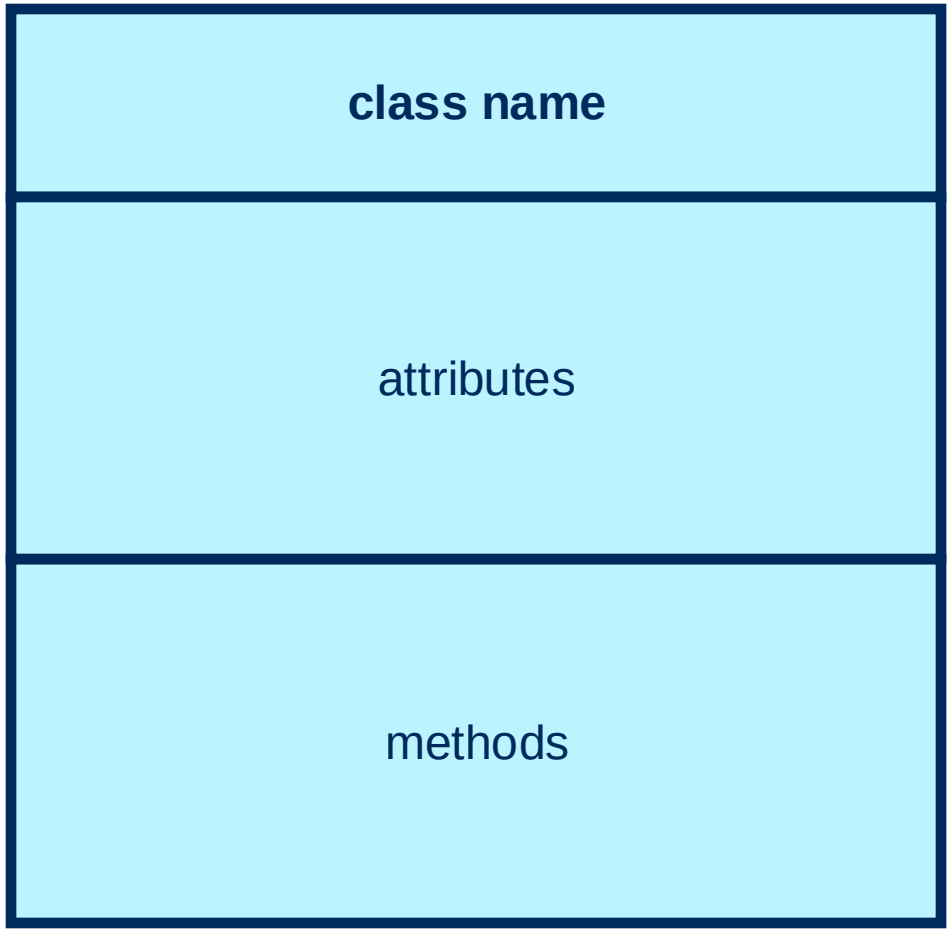
- Be able to analyze a written problem specification and identify a set of classes that describe the data.
- Understand how nouns and verbs relate to objects and methods.

SPECIFICATION ANALYSIS

- Part of the Object-Oriented development paradigm.
- Highly interactive and human-intensive.
- Result: initial set of classes that can represent the data from the problem domain.
- *What* rather than *How*.

THE UNIFIED MODELING LANGUAGE (UML) CLASS DIAGRAMS

- We're going to do a noun-verb analysis and create a bunch of these.
- We're also going to link them up using arrows to represent dependencies.
- Further reading: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/



THE PROCESS

- Typically, you are given a specification (a written description of the requirements) for the software system.
- You work in a team.
- Ideally, you all gather around a table.
- You might use index cards, one per class, so you can move them around.

DISCOVERING CLASSES

- Read the specification. Again. And again. The first goal is to figure out how to represent the data in your program.
- Identify core **classes** (simplistic advice: look for important nouns and noun phrases).
- Identify **methods** (simplistic advice: look for verbs).
- Identify **attributes** (simplistic advice: look for adjectives and nouns that describe the core classes).
- Identify other classes that this class needs to talk to in order to fulfil its responsibilities
  - These are its **collaborators**.
- Refine by identifying abstract classes, inheritance, etc.
- Keep adding/refining until everyone on the team is sufficiently satisfied.

EXAMPLE: UNIVERSITY PERSONNEL

Consider this description of a software system that you are developing to keep track of employees and students at a university.

A university wants an application to manage ID cards for people who work for or who attend the university. When a person is hired, they are assigned a unique identifier that we'll call a utor ID. A person may have an arbitrary number of names — first and last, maybe a middle name, maybe just first, maybe seven names. People should be able to change any of their names.

Some people at the university are students. Students also have another unique identifier called a student ID.

When a person's information is displayed, the university wants the full name plus any identifiers, all on one line.

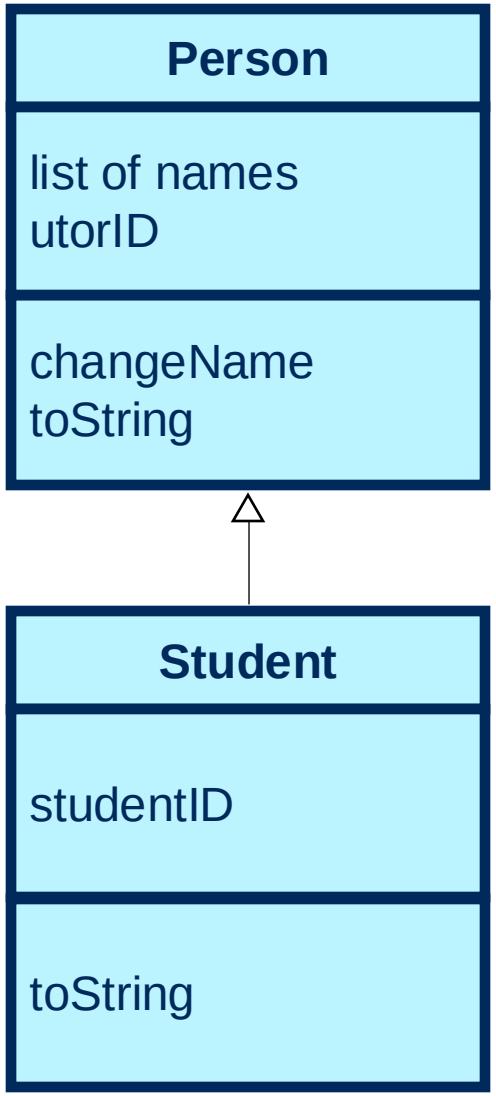
EXAMPLE: UNIVERSITY PERSONNEL

Consider this description of a software system that you are developing to keep track of employees and students at a university.

*A university wants an application to manage ID cards for people who work for or who attend the university. When a **person** is hired, they are assigned a unique identifier that we'll call a utor ID. A person may have an arbitrary number of names — first and last, maybe a middle name, maybe just first, maybe seven names. People should be able to change any of their names.*

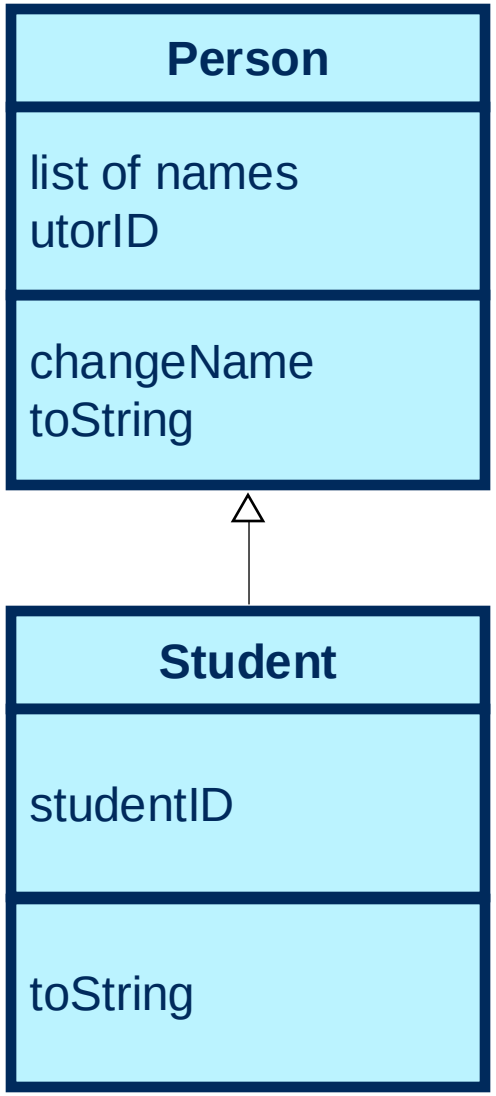
*Some people at the university are **students**. Students also have another unique identifier called a student ID.*

When a person's information is displayed, the university wants the full name plus any identifiers, all on one line.


Person
list of names
utorID
changeName
toString
Student
studentID
toString

An open arrowhead indicates inheritance.

A solid arrowhead indicates composition.



Thoughts on what types these should be?

Later, we'll need to decide on parameters and return types

What type?

Does Student need its own toString method? Why or why not?

RECALL OUR KEY STEPS

Identify important nouns.

- Underline nouns that may make sensible classes or that describe information a class could be responsible for storing.

Choose potential classes.

- From the nouns identified, write down the ones that are potential classes.

Identify verbs that describe responsibilities.

- In the problem description, circle verbs that describe tasks that a class may be responsible for doing.

Identify the properties of the important nouns.

- In the problem description, some nouns describe properties of important nouns. These are the instance variables.

PRACTICE

Each restaurant corresponds to a certain price range, neighbourhood, and cuisines it serves. Restaurants that serve alcohol must have a license, which they need to renew every year. The system should also report how long, on average, customers wait for take out in restaurants that offer take-out service.

When reviewers leave a review for a restaurant, they must specify a recommendation (Thumbs Up or Thumbs Down) and can also leave a comment. An owner of a restaurant can respond to a review with a comment. All users of the system log in with their username. Users can choose to be contacted by email ...