# Inheritance
# Part 1

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY | EECS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

# What is Inheritance?

- A thing sometimes can be described as a specialized type of another thing
  - E.g., a car is a particular type of vehicle
  - E.g., a dog is a particular type of animal
  - E.g., a laptop is a particular type of computer
  - E.g., a cell phone is a particular type of telephone
- Similarly, a class sometimes can be described as an extension or abstraction of another class
- The extended class (child) inherits all the features of the original class (parent) and can implement new/different features for its particular purpose

# Definition and Terminology

- Child (class) = Subclass
- Parent (class) = Superclass
- When *C* inherits from *P*, every feature of *P* is in *C*
- "*C* inherits from *P*" = "*C* extends *P*"
- Inheritance = "*is-a*" relationship = specialization
- Inheritance hierarchy: (graphical) organization of classes related by inheritance

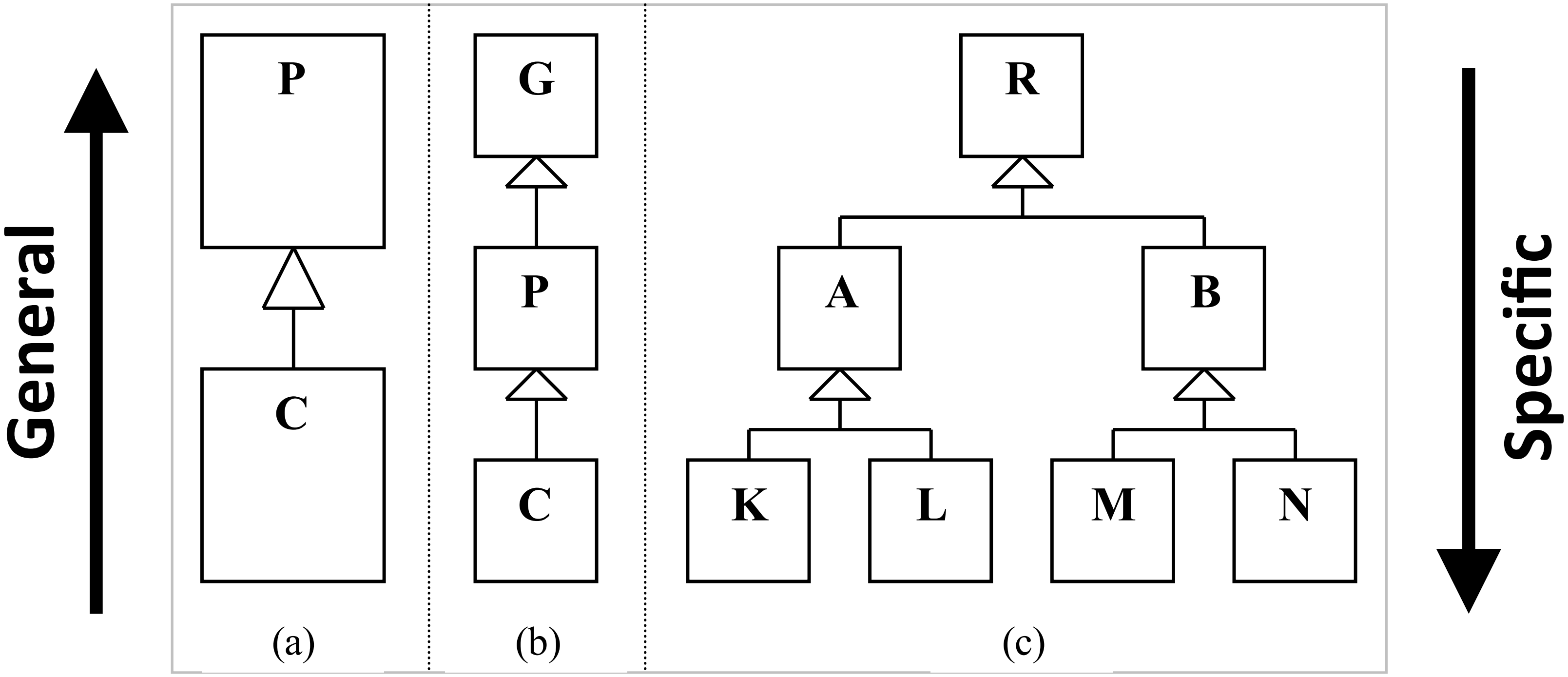
# Benefits of Inheritance

- Code reuse means less code to write

- The parent class has likely already been tested (and passed)

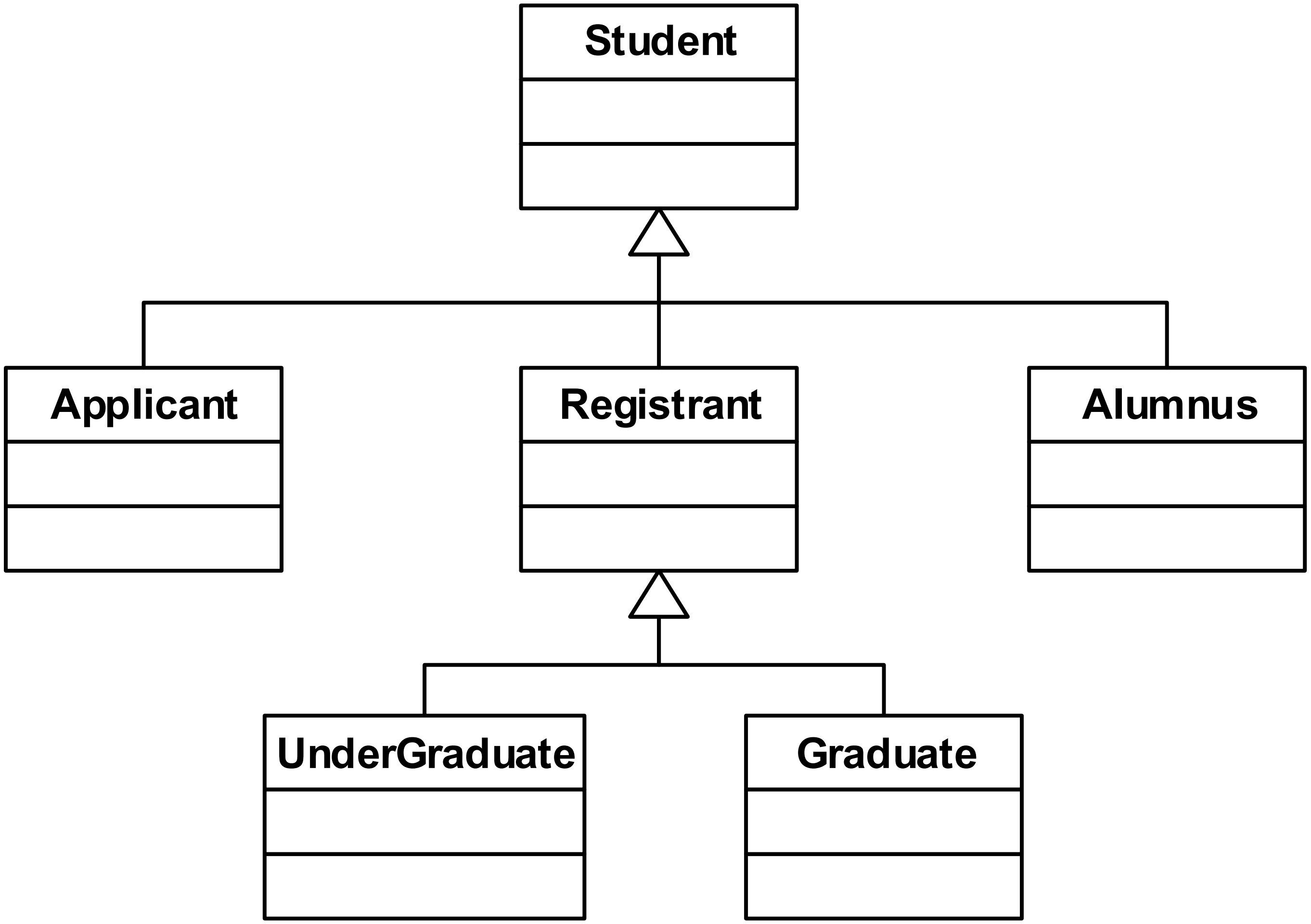
- Changes (fixes or added features) automatically available in the child class

# UML Representation



(a) (b) (c)

# Student Inheritance Hierarchy



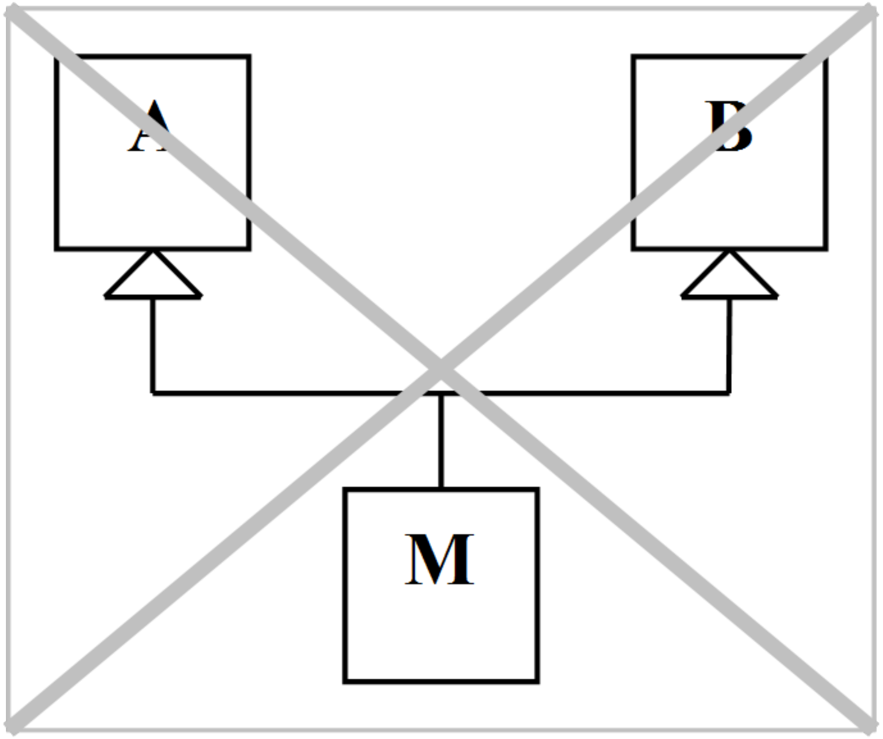
# How to Inherit

- Use the keyword extends in the class declaration to indicate the parent class

```
public class SubClass extends ParentClass
{
       …
```

- Can only extend one class (only one parent)

# No Multiple Inheritance



# Protected Access

- Public access:
  - All classes can access this feature
- Private access:
  - Only accessible to the class in which it is defined
- Protected access:
  - Only accessible to the class in which it is defined and all its derived subclasses (i.e., all its descendants)

# Overriding Methods

- Child class sometimes requires a method with specialized implementation to take advantage of features not applicable to the parent class

- Overriding:
  - Child class keeps parent method's signature and return type

- Not to be confused with overloading:
  - Overloaded methods have same name but different set of arguments and/or return type

# Referring to the Superclass

- Child class still has access to features of parent class using the keyword `super`
- Accessing superclass constructor:
  - `super(`*`anyNecessaryArgs`*`)`
  - Must be the first line in the subclass's constructor
- Accessing superclass method:
  - `super.methodName(`*`anyNecessaryArgs`*`)`

# Examples

- Code available on course website
  - ChequingAccount
  - RewardAccount

- Code demonstrated in lecture