SOFTWARE ENGINEERING

(15B11CI513)

Credits :- 4 **Contact Hours :-** 3-1-0

Introduction to Design and
Unified Modeling Language (UML)

Problem Statement

MelbX, a famous Australian University, has committed to a new registration system for the students, employees and academic staff. The university would like to keep records for each type of university member in a centralized filing system. They would like to implement the system using the latest software development technologies. The system should be able to support the following high level requirements.

1. Keep records for all university members.
2. Capability to add new records to the system.
3. Capability to edit specific records.
4. Capability to enrol/unenrol students in subjects.
5. Capability to assign/unassign course to staff.
6. Capability to add/modify/list prerequisites for a course.

Solving the Problem

- Step 1 - Write Requirements for the system
  - Should be clearly written and unambiguous
  - Should be implementable
  - Should be testable
  - Is not a part of this course
- Step 2 – Analysis, Design (Modeling)
  - You will be learning in this section.....
- Step 3 – Implementation
  - You already know.....

Analysis, Design and Implementation

- Proper analysis and design (modeling) prior to implementation results in a high quality product.
- Successful projects spend most time on analysis, less on design and even less time on implementation.

Analysis	Design	Implementation

Introduction to UML

- Is a graphical modeling language that can be used represent the artifacts of analysis and design.
- It is a standard that has international support.
- Was developed by Rational Software –
  - Grady Booch, Jim Rumbaugh and Ivar Jacobson

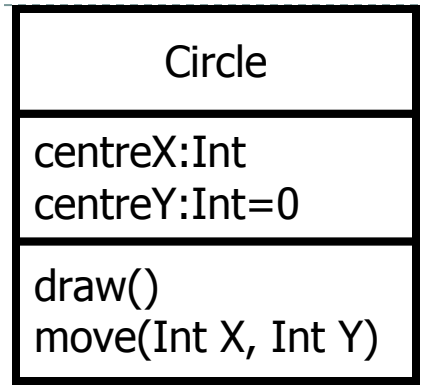
UML™ History

- 1994 – Grady Booch and Jim Rumbaugh started at Rational creating the new notation
  - Grandy Booch – Booch Method
  - Jim Rumbaugh – Object Modeling Technique
- 1995 – Ivar Jacobson Joined the team
  - Object-Oriented Software Engineering
- First Version of UML – 0.8
- 1995 Object Management Group (OMG) agreed to make UML the standard
- 1996 – Additional companies got involved.
- Current version of UML is 2.0

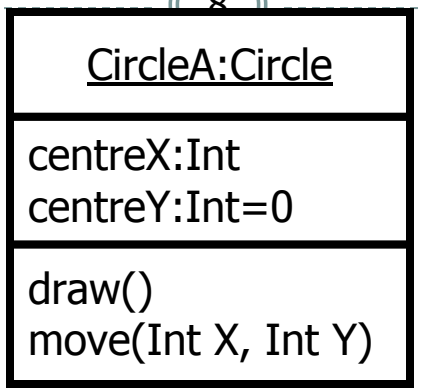
UML™ – Building Blocks

- Elements (Things)
  - *e.g. classes, interfaces etc*
- Relationships
  - *e.g. association, generalization*
- Diagrams
  - *e.g. class diagrams, use case diagrams*

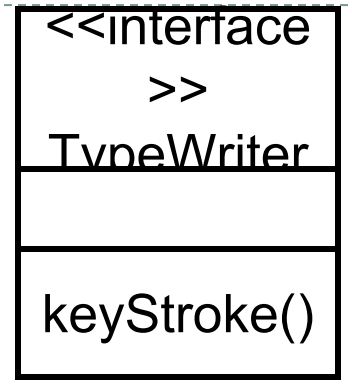
UML™ - Elements



Class

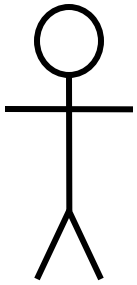


Object

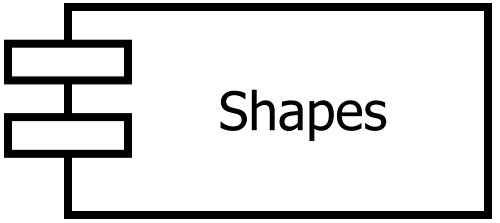


Interface

Use-case



Actor



Component

UML™ - Relationships

- Dependency

- Association

- Generalization

- Realization

UML™ - Diagrams

- Structural Diagrams
  - Visualize, specify, construct and document the *STATIC* aspects of the system.
- Behavioral Diagrams
  - Visualize, specify, construct and document the *DYNAMIC* aspects of the system.

UML™ – Diagrams – cont..

Structural
Class Diagram
Object Diagram
Component Diagram
Deployment Diagram

Behavioral
Use case Diagram
Sequence Diagram
Collaboration Diagram
Statechart Diagram
Activity Diagram

UML™ – Diagrams - Notes

- **Class Diagram** shows a set of classes, interfaces and their relationships.
- **Object Diagram** shows a set of objects and their relationships.
- **Component Diagram** shows the organization and dependencies among a set of of components.
- **Deployment Diagram** shows the configuration of run time processing nodes.

UML™ – Diagrams - Notes

- **Use Case Diagram** shows use cases, actors and their relationships.
- **Sequence Diagram** shows the interaction between actors and objects and other objects of the system in ordered based on the time they occur.
- **Collaboration Diagram** shows the structural organization of the objects that send and receive messages.
- **Statechart Diagram** shows a state machine consisting of states, transitions an events.
- **Activity Diagram** used to analyze the bahaviour within more complex use-cases.

Life Cycle Phase  Diagrams

- Requirements
- Object Oriented Analysis
- Object Oriented Design
- Development
- Deployment

Analysis ⟷ Diagrams

Activity	UML Diagram
Understand System Usage	Use-case Diagram
Define Workflows	Activity Diagram
Identify Classes	High Level Class Diagram

Design  Diagrams

Activity	UML Diagram
Identify Interactions among objects	Sequence and Collaboration Diagram
Analyze State Changes	State Diagram
Refine Class Diagrams	Class Diagram

UML Modeling Tools

- Rational Rose
  - Provides complete UML support
  - As many useful capabilities
  - Provides complete object management
- Visio
- Many other free tool are available on the web –
  - Be careful since some of them don't allow printing.
- Pen and Paper

Use Case Diagrams

Introduction

- Getting started is the most difficult part of any new process.
- In software modelling, the first thing you need to do is understand what are you going to model and ultimately develop.
- Creating a highest form details about a system--use case diagram--is an almost natural point of origin for the software design.
- A use case diagram is an excellent way to communicate to management, customers, and other non-development people what a system will do when it is completed.

University Record System (URS)

- A University record system should keep information about its students and academic staff.
- Records for all university members are to include their id number, surname, given name, email, address, date of birth, and telephone number.
  - Students and academic staff each have their own unique ID number: studN (students), acadN (academic employee), where N is an integer (N>0).
- In addition to the attributes mentioned above:
  - Students will also have a list of subjects they are enrolled in. A student cannot be enrolled in any more than 10 subjects.
  - Academic employees will have a salary, and a list of subjects they teach. An academic can teach no more than 3 subjects.

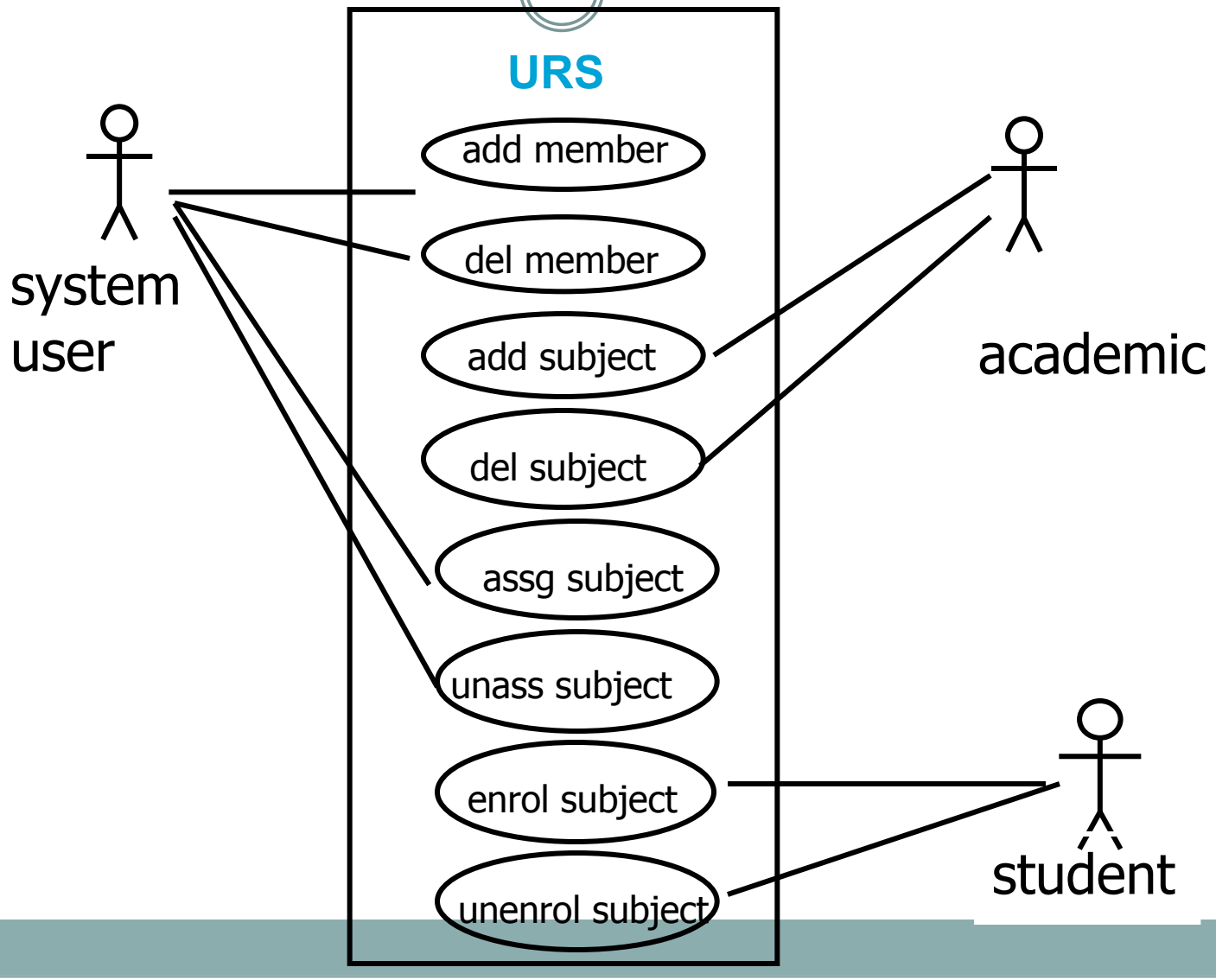
Some Actions Supported by URS

- The system should be able to handle the following commands.
  - Add and remove university members (students, and academic staff)
  - Add and Delete subjects
  - Assign and Un-assign subjects to students
  - Assign and Un-assign subjects to academic staff.

Use Case Diagrams

- Use Case diagrams show the various activities the users can perform on the system.
  - System is something that performs a function.
- They model the dynamic aspects of the system.
- Provides a *user's* perspective of the system.

Use Case Diagram - URS System

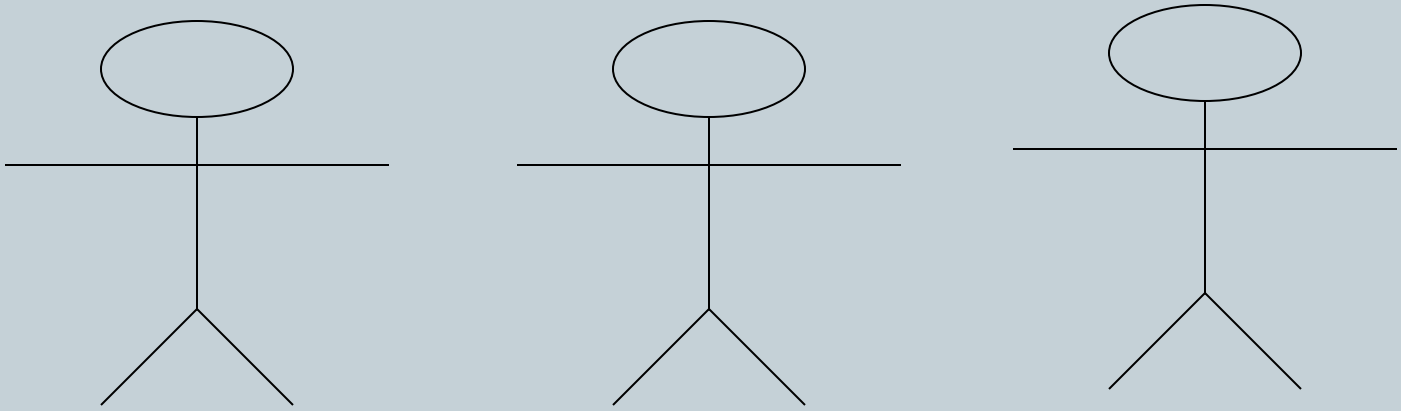


Use Case Diagrams

- A set of *ACTORS* : roles users can play in interacting with the system.
  - An actor is used to represent something that users our system.
- A set of *USE CASES*: each describes a possible kind of interaction between an actor and the system.
  - Uses cases are actions that a user takes on a system
- A number of *RELATIONSHIPS* between these entities (Actors and Use Cases).
  - Relationships are simply illustrated with a line connecting actors to use cases.

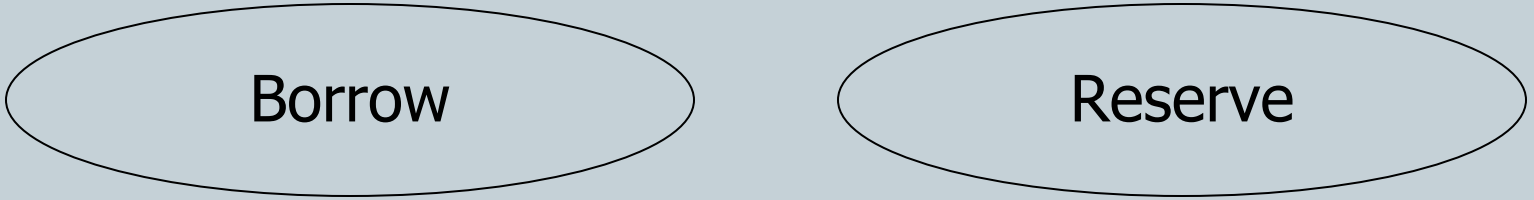
Use Case Diagrams - Actors

- An *actor* is a user of the system playing a particular role.
- Actor is shown with a stick figure.

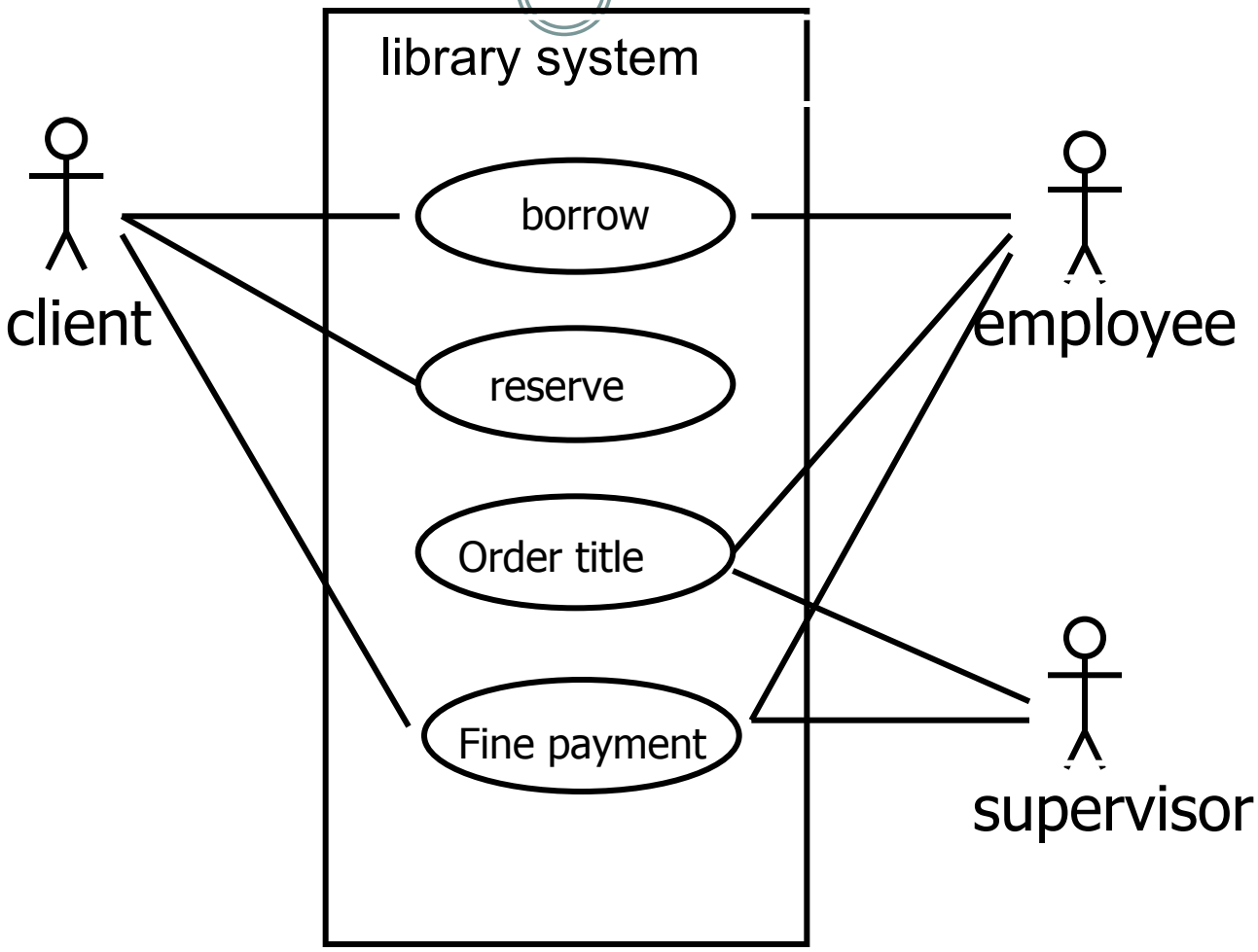


Use Case Diagrams – Use Cases

- Use case is a particular activity a user can do on the system.
- Is represented by an ellipse.
- Following are two use cases for a library system.

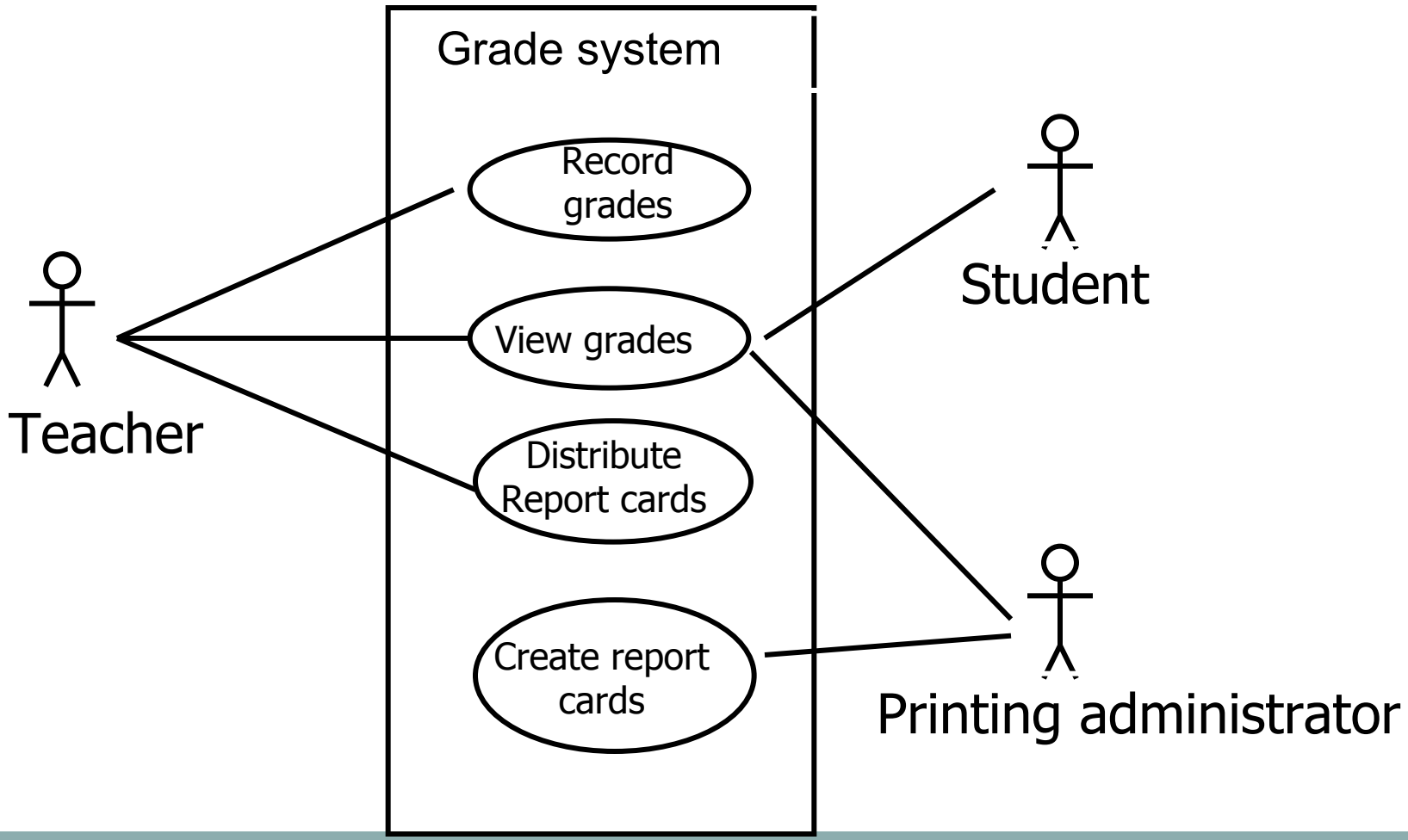


Use Case Diagram – Example1 (Library)



A Library System.

Use Case Diagram for Student Assessment Management System



Use Case Vs Scenarios

- Each use case is one or more scenarios.
  - Add Subject Use Case :
    - Scenario 1 : Subject gets added successfully.
    - Scenario 2 : Adding the subject fails since the subject is already in the database.
  - Enroll Subject Use Case:
    - Scenario 1 : Student is enrolled for the subject.
    - Scenario 2 : Enrollment fails since the student is already enrolled in the subject.
- Each scenario has a sequence of steps.

Scenarios

- Each scenario has a sequence of steps.
  - Scenario 1 : Student is enrolled for the subject.
    - Student chooses the “enroll subject” action.
    - Check the student has enrolled in less than 10 subjects.
    - Check if the subject is valid.
    - Assign the subject to the student.

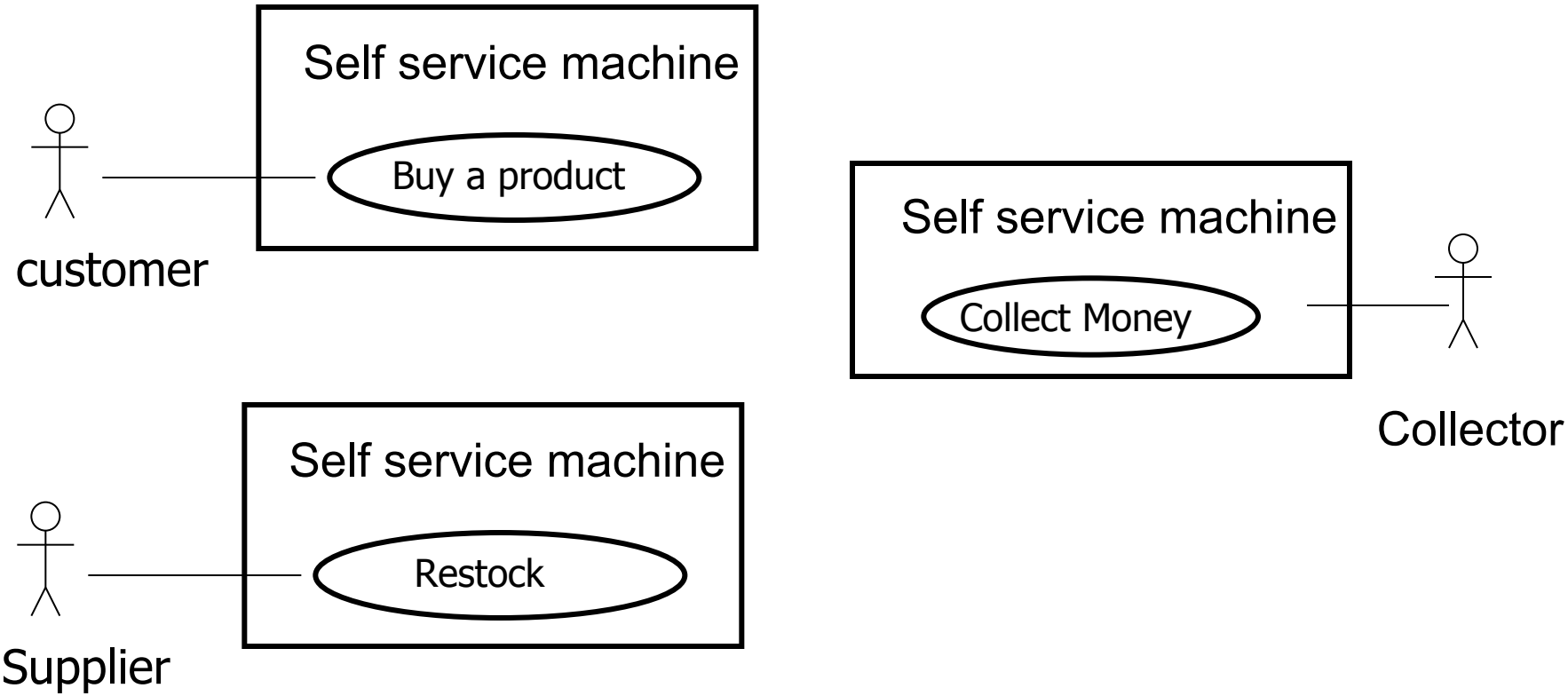
Scenarios

- Each scenario has a sequence of steps.
  - Scenario 2 : Enrolling fails since the student is already enrolled in 10 subjects.
    - Student chooses the “enroll subject” action.
    - Check the student has enrolled in less than 10 subjects.
    - Return an error message to the student.

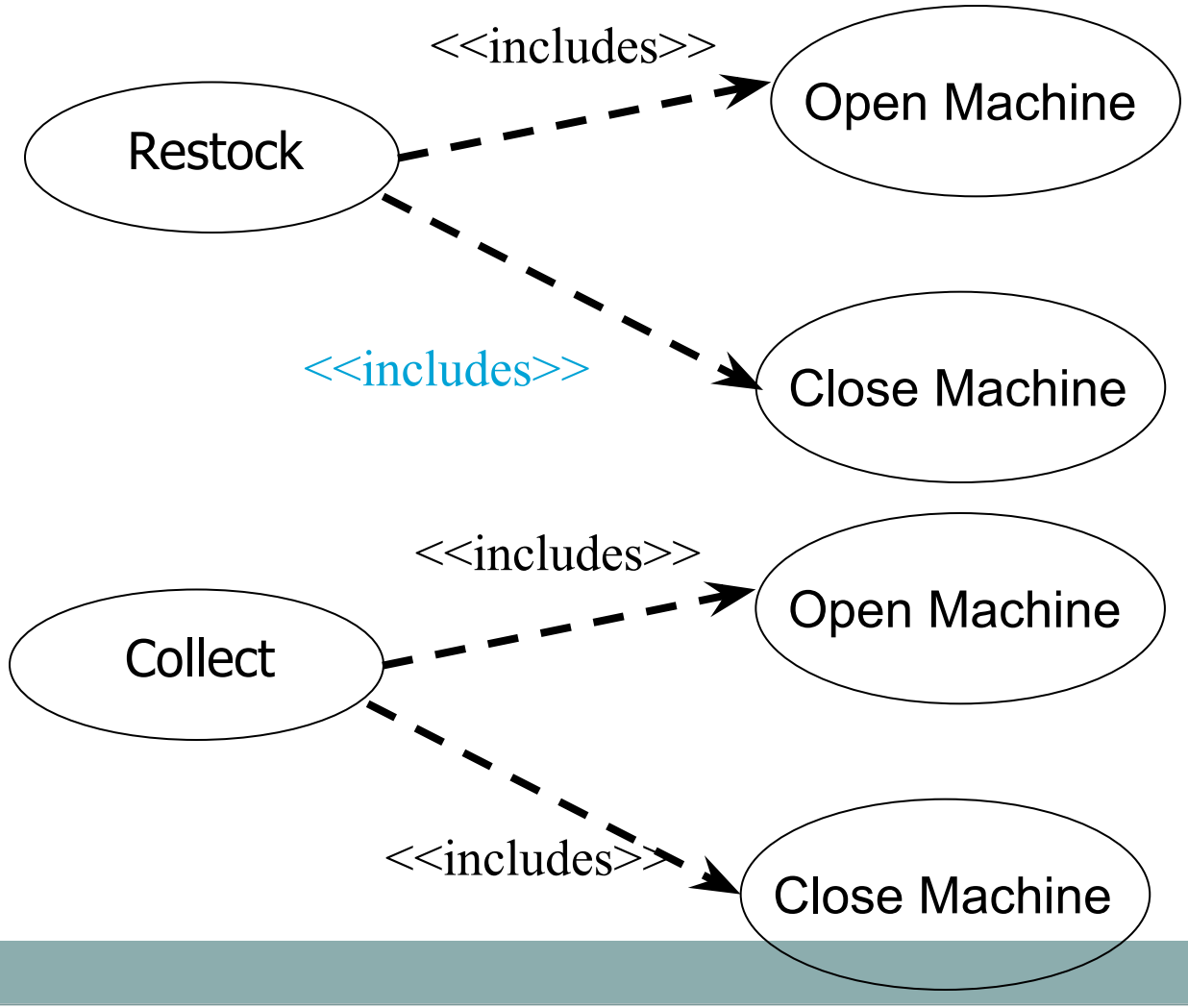
Use Case Diagrams - Relationships

- Inclusion
  - *Inclusion enables to reuse one use case's steps inside another use case.*
- Extension
  - *Allows creating a new use case by adding steps to existing use cases*
- Generalization
  - *Allows child use cases to inherit behavior from parent use cases*

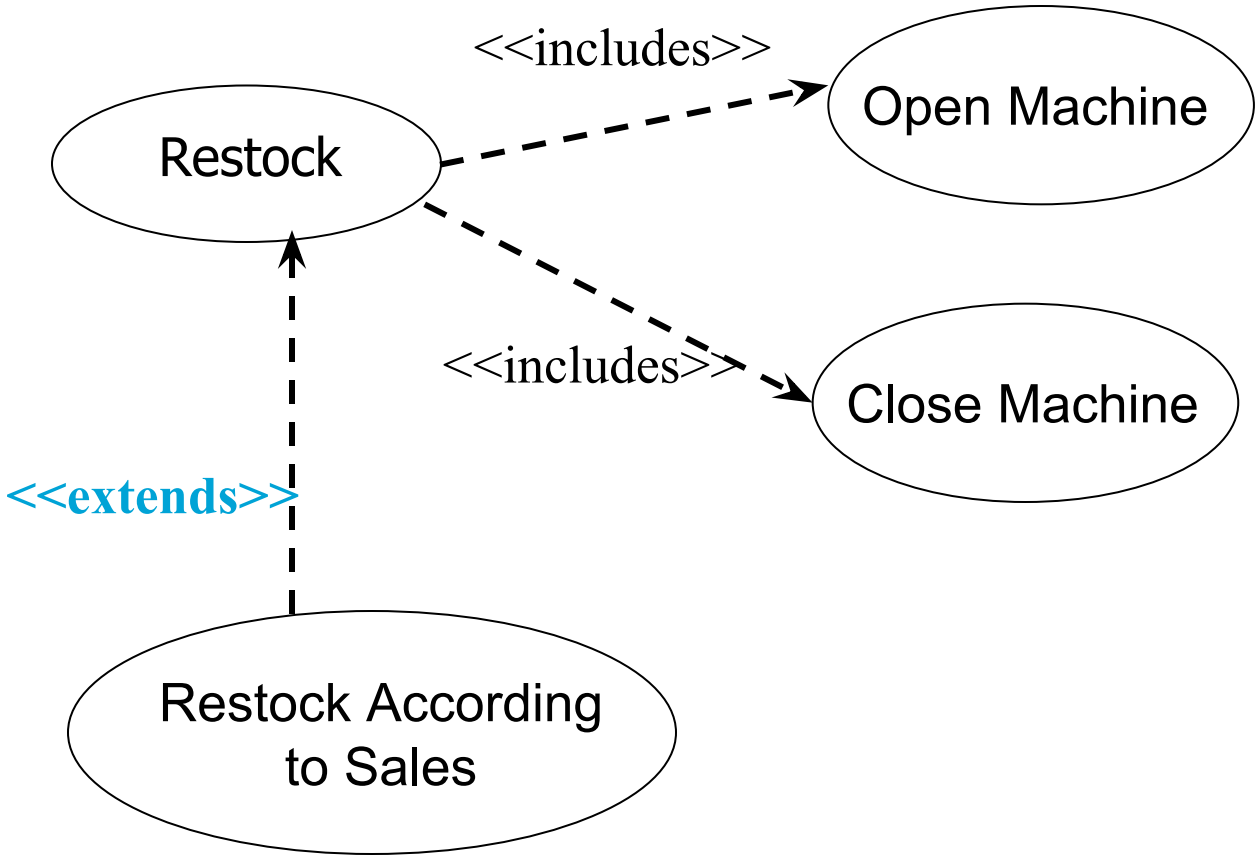
Use Case – Example (self service machine)



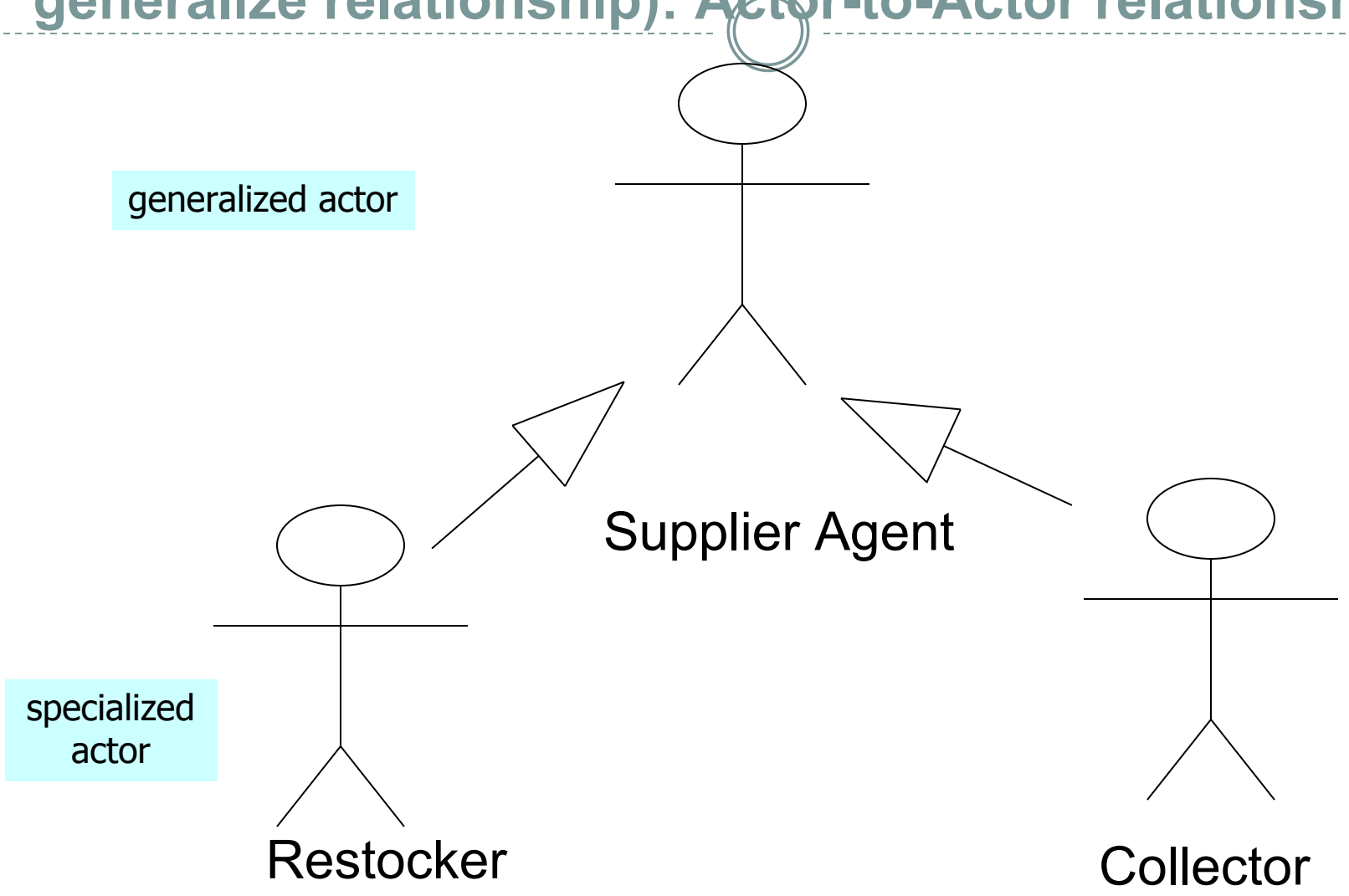
Use Case – Example (self service machine – includes relationship)



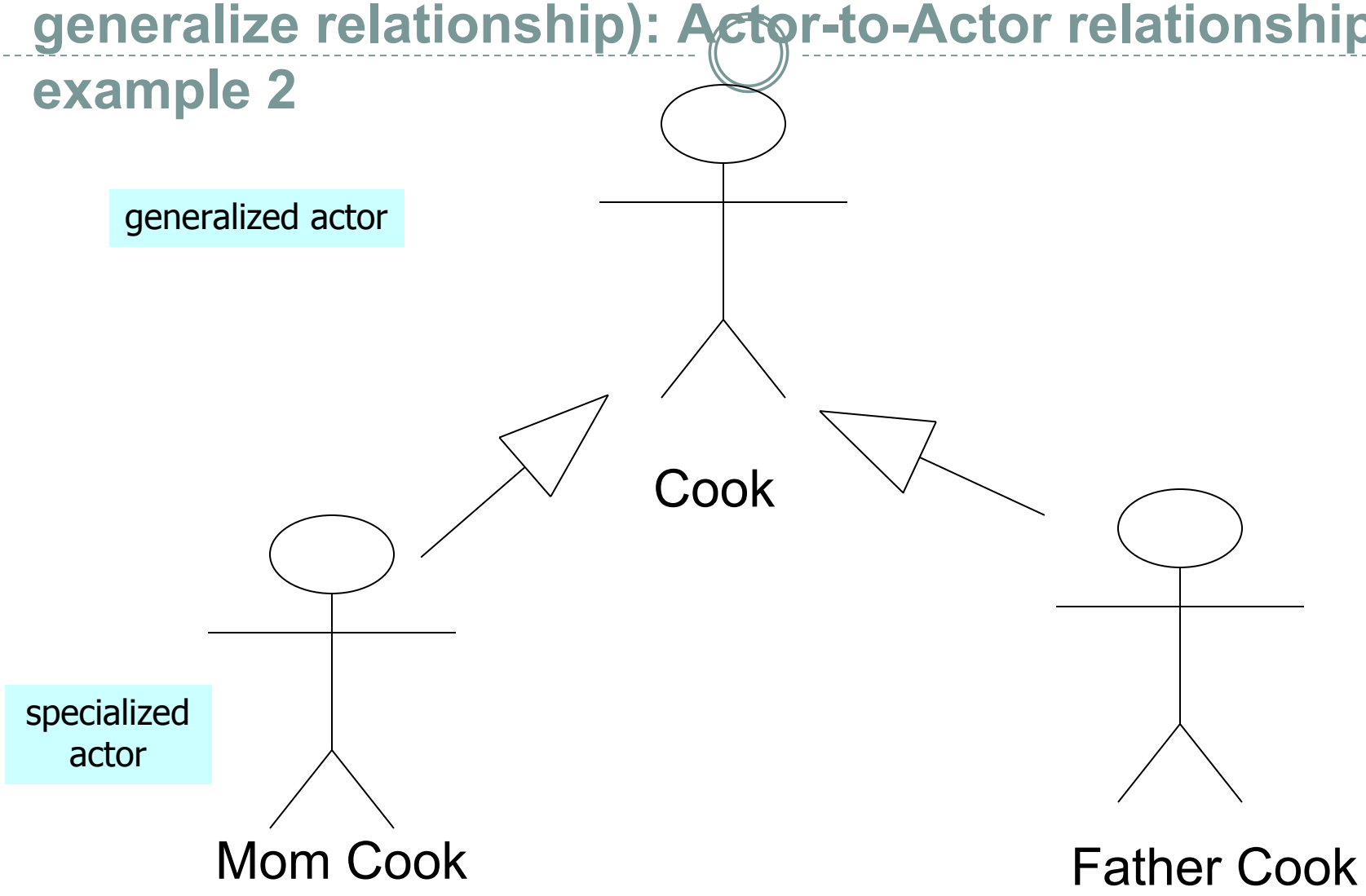
Use Case – Example (self service machine – extends relationship)



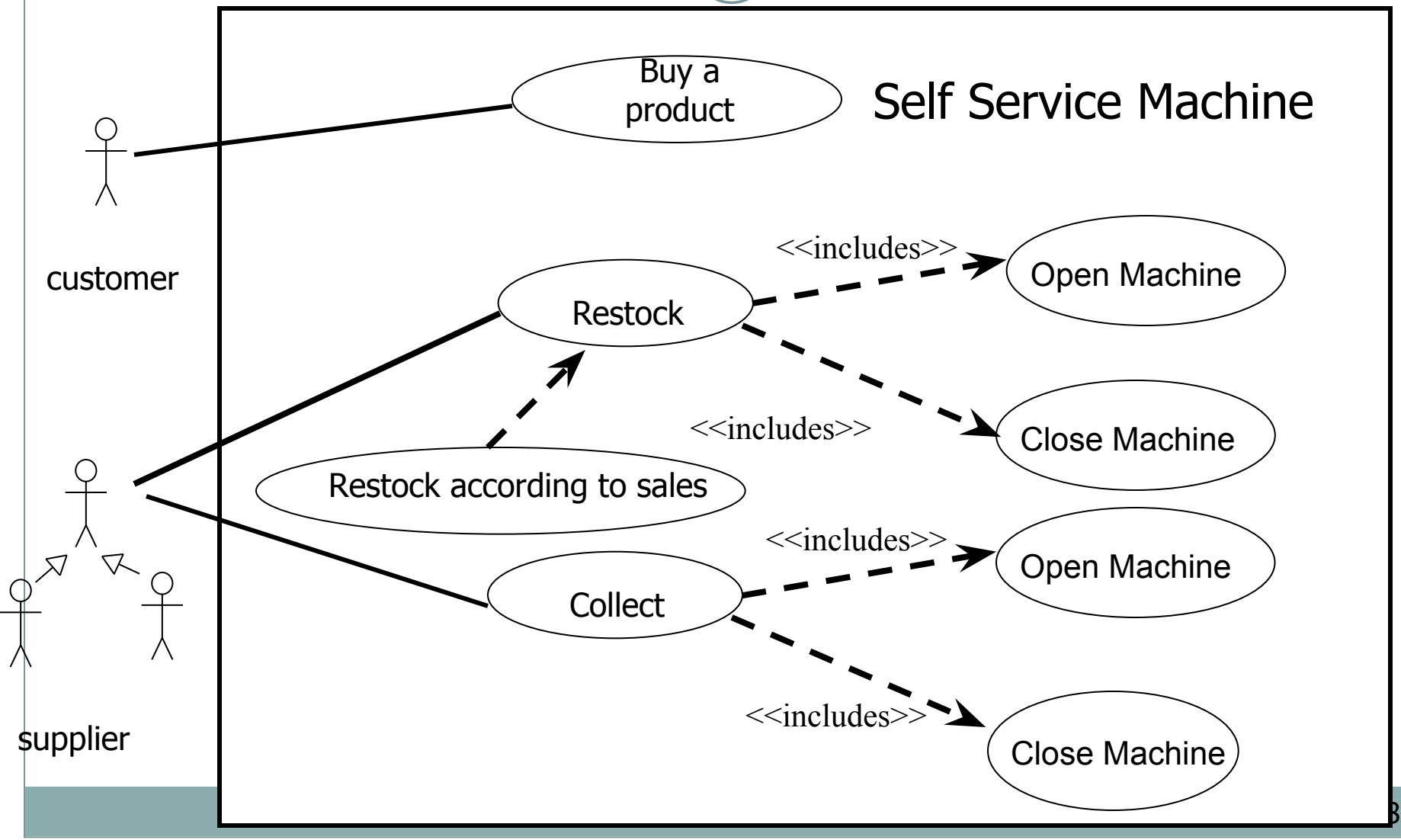
Use Case – Example (self service machine – generalize relationship): Actor-to-Actor relationship



Use Case – Example (self service machine – generalize relationship): Actor-to-Actor relationship – example 2



Use Case – Example (self service machine)



From Use Case to Classes

Identify Classes (Extract Nouns)

- A **University** **record system** should keep information about its **students** and **academic staff.**
- **Records** for all **university members** are to include their **id number**, **surname**, **given name**, **email**, **address**, **date of birth**, and **telephone number.**
  - Students and academic staff each have their own unique ID number: **studN** (students), **acadN** (academic employee), where N is an integer (N>0).
- In addition to the attributes mentioned above:
  - Students will also have a list of **subjects** they are enrolled in. A student cannot be enrolled in any more than 10 subjects.
  - Academic employees will have a salary, and a list of subjects they teach. An academic can teach no more than 3 subjects.

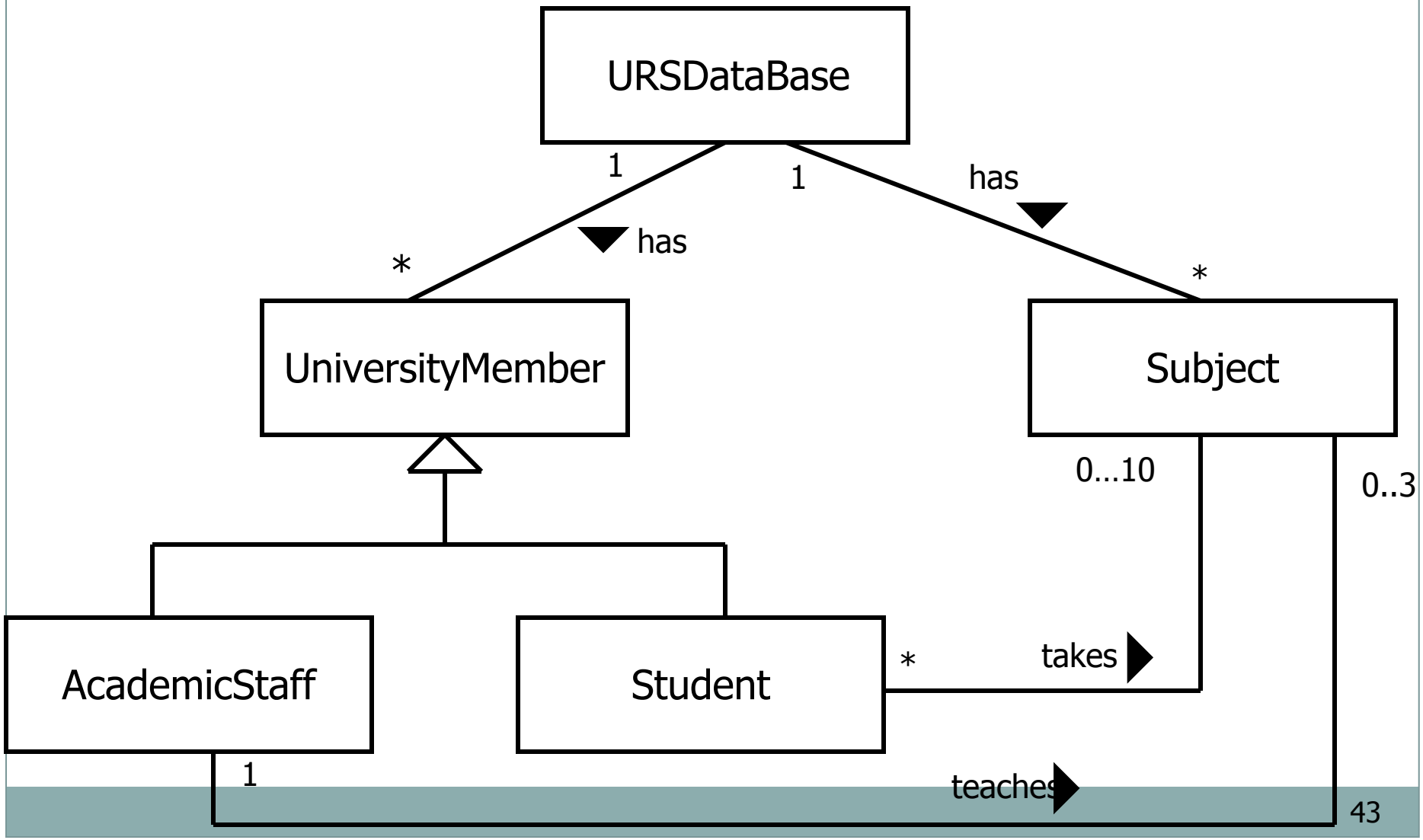
Nouns which are potential classes

- A ***University record system*** should keep information about its ***students*** and ***academic staff***.
- **Records** for all ***university members*** are to include their **id number**, **surname**, **given name**, **email**, **address**, **date of birth**, and **telephone number**.
  - Students and academic staff each have their own unique ID number: **studN** (students), **acadN** (academic employee), where N is an integer (N>0).
- In addition to the attributes mentioned above:
  - Students will also have a list of ***subjects*** they are enrolled in. A student cannot be enrolled in any more than 10 subjects.
  - Academic employees will have a salary, and a list of subjects they teach. An academic can teach no more than 3 subjects.

Classes identified in the first pass

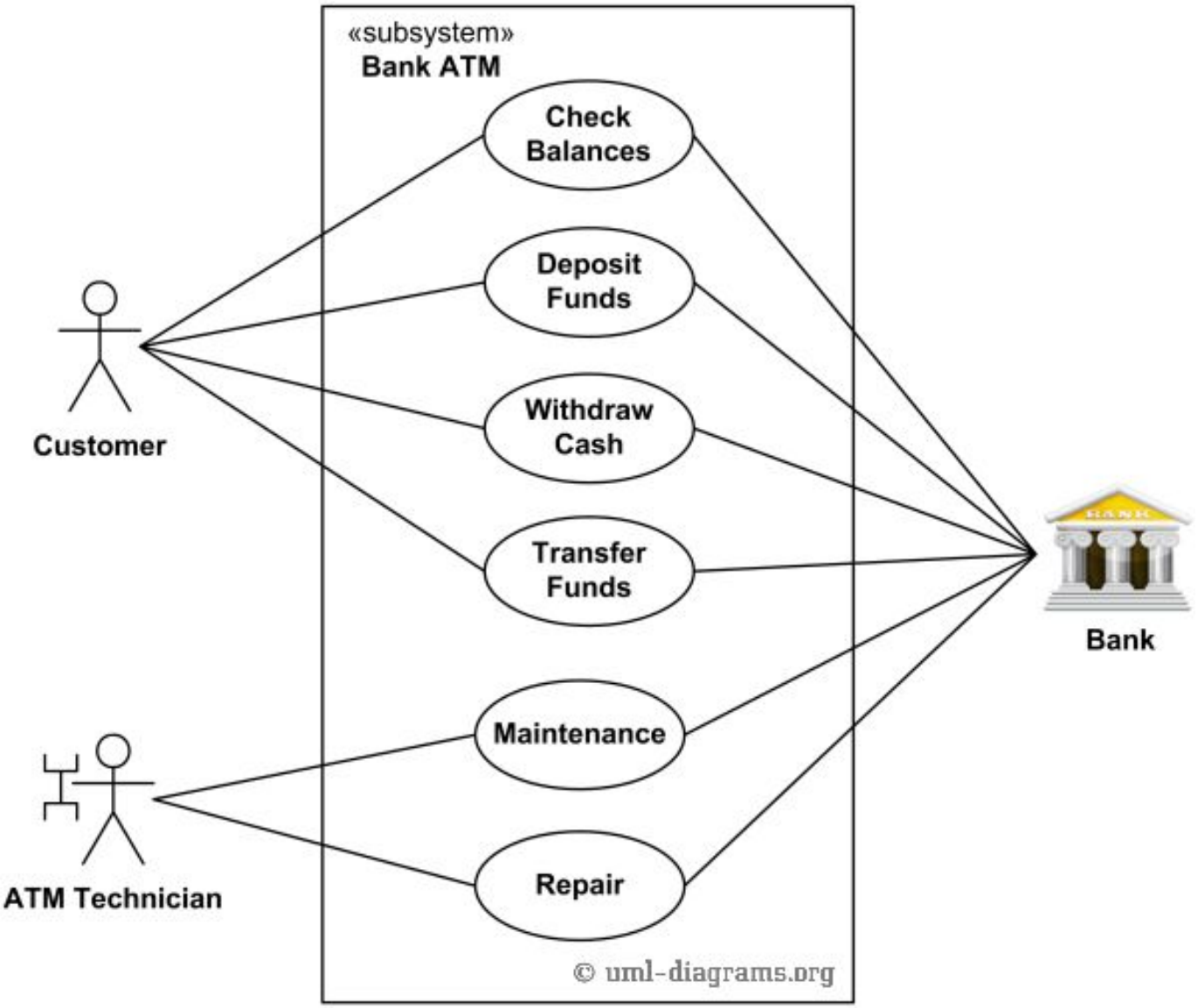
- UniversityRecordSystem - URS
- Student
- Academic Staff
- UniversityMembers
- Subject

URS - High Level Class Diagram



Bank ATM example

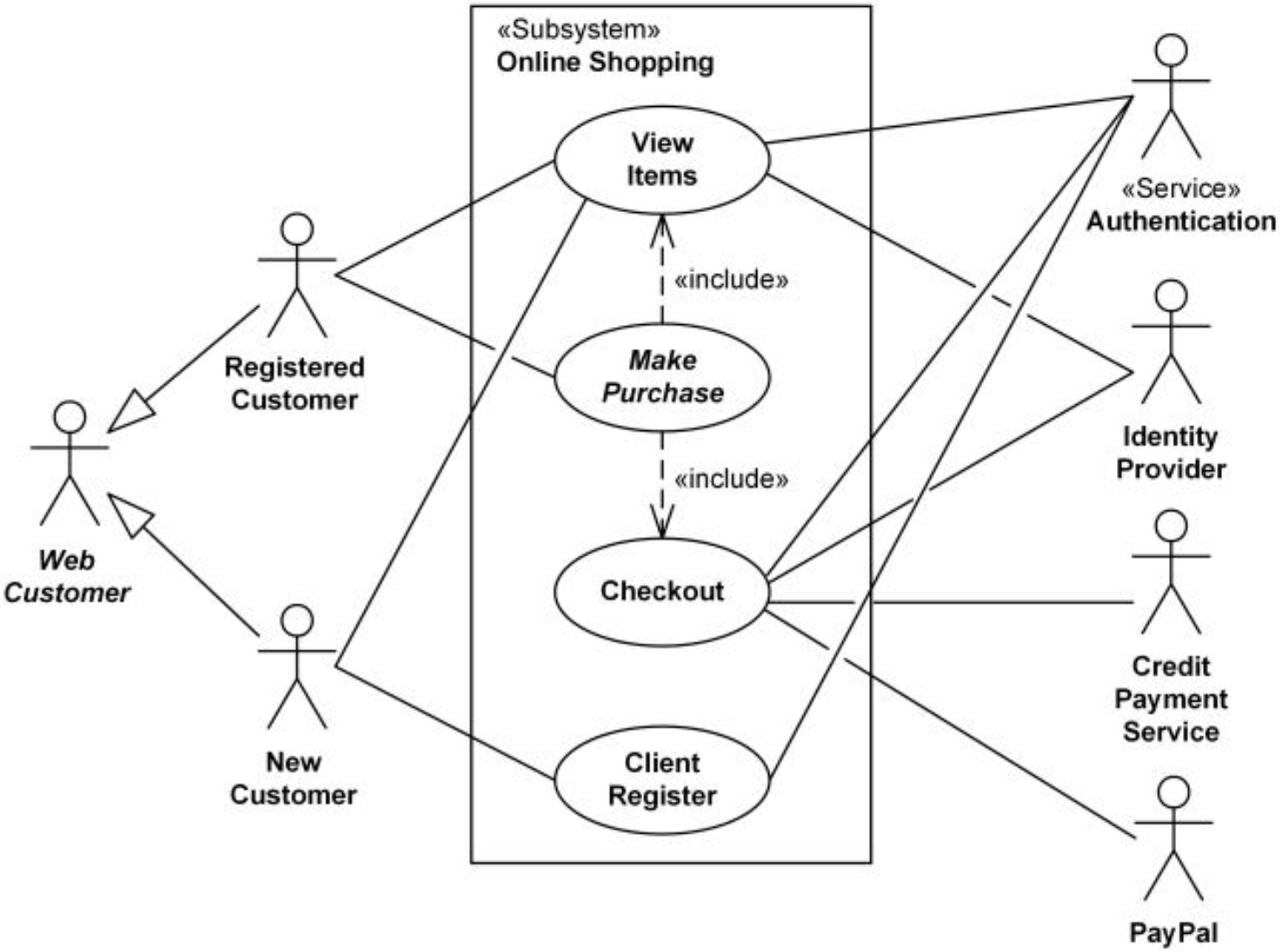
- **Purpose**: Describe use cases that an automated teller machine (ATM) or the automatic banking machine (ABM) provides to the bank customers.
- **Summary**: Customer uses a bank ATM to check balances of his/her bank accounts, deposit funds, withdraw cash and/or transfer funds (use cases). ATM Technician provides maintenance and repairs to the ATM.
-


«subsystem»
Bank ATM
Check Balances
Deposit Funds
Withdraw Cash
Transfer Funds
Maintenance
Repair
Customer
ATM Technician
Bank
© uml-diagrams.org

Online shopping example

- **Purpose**: Provide top level use cases for a web customer making purchases online.

- **Summary**: Web customer actor uses some web site to make purchases online. Top level use cases are **View Items**, **Make Purchase** and **Client Register**.


«Subsystem»
Online Shopping
View Items
«include»
Make Purchase
«include»
Checkout
Client Register
Registered Customer
Web Customer
New Customer
«Service»
Authentication
Identity Provider
Credit Payment Service
PayPal