SFWR ENG 3A04: Software Design II

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Term 1, 2008–2009

Acknowledgments: Material based on *Software Architecture Design* by Tao et al. (Chapter 4)

Outline of Part I

1. UML for Software Architecture

2. Questions???

Outline of Part II

3 Overview

4 OO Analysis and Design
- OO Analysis
- OO Design

Part I

Review of Previous Lecture

Part II

Today's Lecture

OO Analysis and Design Overview

- The Object Oriented (OO) architecture style emerged from the conceptualization of abstract data types (ADT)

OO Analysis and Design Overview

- The Object Oriented (OO) architecture style emerged from the conceptualization of abstract data types (ADT)
- An abstract data type is the specification of a set of logically correlated data items, and the operations that can be performed on them

OO Analysis and Design Overview

- The Object Oriented (OO) architecture style emerged from the conceptualization of abstract data types (ADT)

- An abstract data type is the specification of a set of logically correlated data items, and the operations that can be performed on them

- string, tree, and vector can be regarded as ADT

OO Analysis and Design Overview

- The Object Oriented (OO) architecture style emerged from the conceptualization of abstract data types (ADT)
- An abstract data type is the specification of a set of logically correlated data items, and the operations that can be performed on them
- string, tree, and vector can be regarded as ADT
- The use of ADT allows separation of interface and implementation

Overview

- The Object Oriented (OO) architecture style emerged from the conceptualization of abstract data types (ADT)
- An abstract data type is the specification of a set of logically correlated data items, and the operations that can be performed on them
- string, tree, and vector can be regarded as ADT
- The use of ADT allows separation of interface and implementation
- There can be various concrete implementations for a same ADT

OO Analysis and Design Overview

- The type theory of ADTs is derived by analogy with abstract algebra

Overview

- The type theory of ADTs is derived by analogy with abstract algebra
- An ADT implementation is an algebra

Overview

- The type theory of ADTs is derived by analogy with abstract algebra
- An ADT implementation is an algebra
- As in algebra, the exact nature of the set is considered to be less important than the relative effects of the different operations

Overview

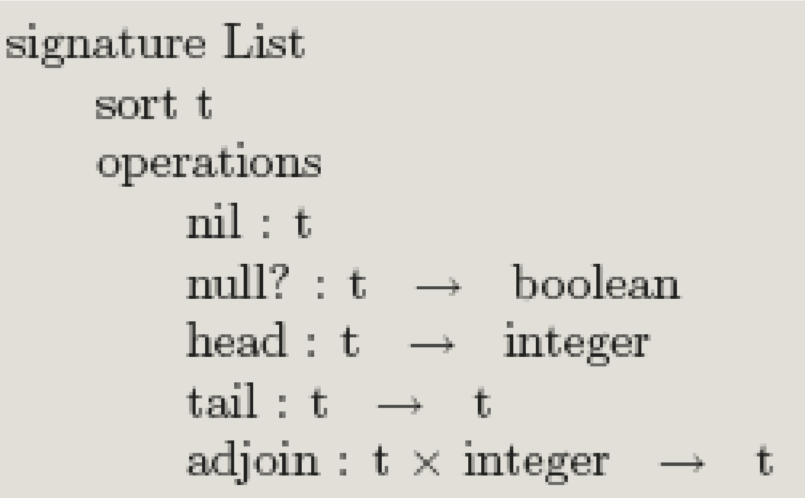
- The type theory of ADTs is derived by analogy with abstract algebra
- An ADT implementation is an algebra
- As in algebra, the exact nature of the set is considered to be less important than the relative effects of the different operations
- The operations in an algebra is characterized by its signature

- The type theory of ADTs is derived by analogy with abstract algebra
- An ADT implementation is an algebra
- As in algebra, the exact nature of the set is considered to be less important than the relative effects of the different operations
- The operations in an algebra is characterized by its signature
- A signature provides a name for the carrier set, or sort, of the algebra, and specifies the operations on the carrier

Overview

- The type theory of ADTs is derived by analogy with abstract algebra
- An ADT implementation is an algebra
- As in algebra, the exact nature of the set is considered to be less important than the relative effects of the different operations
- The operations in an algebra is characterized by its signature
- A signature provides a name for the carrier set, or sort, of the algebra, and specifies the operations on the carrier

OO Analysis and Design Overview



```
signature List
    sort t
    operations
        nil : t
        null? : t  →  boolean
        head : t  →  integer
        tail : t  →  t
        adjoin : t × integer  →  t
```

Overview

- Such separation (due to abstraction) allows great flexibility in software analysis, design, and development

OO Analysis and Design Overview

- Such separation (due to abstraction) allows great flexibility in software analysis, design, and development
- Senior software analysts and architects can concentrate on the abstract and logical design of the system

OO Analysis and Design Overview

- Such separation (due to abstraction) allows great flexibility in software analysis, design, and development
- Senior software analysts and architects can concentrate on the abstract and logical design of the system
- Programmers can furnish the details of each component in its implementation

Overview

- Such separation (due to abstraction) allows great flexibility in software analysis, design, and development
- Senior software analysts and architects can concentrate on the abstract and logical design of the system
- Programmers can furnish the details of each component in its implementation
- Result: greatly reduces the complexity of software development and help to increase the reliability of the whole system

OO Analysis and Design Overview

There are generally three main OO principles:

- Encapsulation:

There are generally three main OO principles:

- Encapsulation:
    - Outside world of a class only sees its functional specification (not how it is implemented)

There are generally three main OO principles:

- Encapsulation:
  - Outside world of a class only sees its functional specification (not how it is implemented)
  - It allows the flexibility of design (public interface of a class has to remain intact)

OO Analysis and Design Overview

There are generally three main OO principles:

- Encapsulation:
  - Outside world of a class only sees its functional specification (not how it is implemented)
  - It allows the flexibility of design (public interface of a class has to remain intact)
- Inheritance:

OO Analysis and Design Overview

There are generally three main OO principles:

- Encapsulation:
  - Outside world of a class only sees its functional specification (not how it is implemented)
  - It allows the flexibility of design (public interface of a class has to remain intact)
- Inheritance:
  - Programmers can define derived classes from base classes to inherit existing attributes and operations

OO Analysis and Design Overview

There are generally three main OO principles:

- Encapsulation:
    - Outside world of a class only sees its functional specification (not how it is implemented)
    - It allows the flexibility of design (public interface of a class has to remain intact)
- Inheritance:
    - Programmers can define derived classes from base classes to inherit existing attributes and operations
    - Allows re-using existing software components (cost)

OO Analysis and Design Overview

- Polymorphism:

Overview

- Polymorphism:
  - poly means many and morph means form

OO Analysis and Design Overview

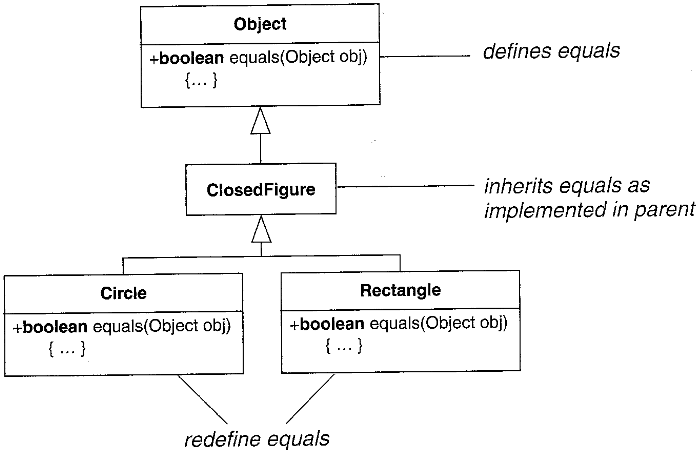
- Polymorphism:
  - poly means many and morph means form
  - Refers to the ability of an object to behave differently and assume many different forms

Overview

- Polymorphism:
    - poly means many and morph means form
    - Refers to the ability of an object to behave differently and assume many different forms
    - Includes vertical override operations between parent classes and derived classes

Overview

- Polymorphism:
  - poly means many and morph means form
  - Refers to the ability of an object to behave differently and assume many different forms
  - Includes vertical override operations between parent classes and derived classes
  - Includes horizontal overloading operations within the same class


Object
+boolean equals(Object obj)
{… }
defines equals
ClosedFigure
inherits equals as implemented in parent
Circle
+boolean equals(Object obj)
{ … }
Rectangle
+boolean equals(Object obj)
{ … }
redefine equals

Overview

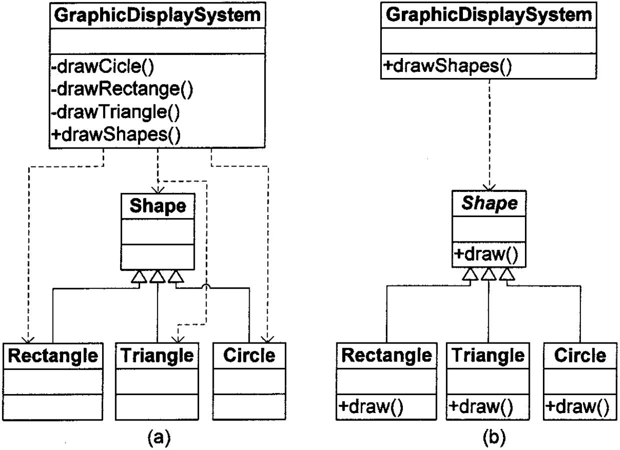


Figure: Vertical override operation (Used for decoupling)

- Polymorphism (Continued):

OO Analysis and Design Overview

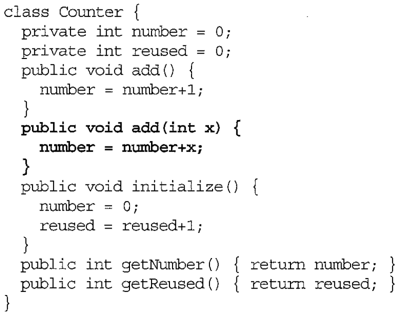
- Polymorphism (Continued):
  - Includes horizontal overloading operations within the same class

OO Analysis and Design   Overview

- Polymorphism (Continued):
    - Includes horizontal overloading operations within the same class
        - Attribute names of a class may be the same as those in another class since they are accessed independently (different scopes)

- Polymorphism (Continued):
  - Includes horizontal overloading operations within the same class
    - Attribute names of a class may be the same as those in another class since they are accessed independently (different scopes)
    - Within a Java class construct, methods may share the same name as long as they may be distinguished either by the number of parameters, or different parameter types (signature)

OO Analysis and Design Overview



```
class Counter {
  private int number = 0;
  private int reused = 0;
  public void add() {
    number = number+1;
  }
  public void add(int x) {
    number = number+x;
  }
  public void initialize() {
    number = 0;
    reused = reused+1;
  }
  public int getNumber() { return number; }
  public int getReused() { return reused; }
}
```

Figure: Horizontal overloading operation

OO Analysis and Design OO Analysis and Design

- Generally, an OO development process has three stages:

OO Analysis and Design

- Generally, an OO development process has three stages:

  1. OO analysis

OO Analysis and Design

- Generally, an OO development process has three stages:
  1. OO analysis
  2. OO design

OO Analysis and Design

- Generally, an OO development process has three stages:
  1. OO analysis
  2. OO design
  3. OO implementation

OO Analysis and Design

- Generally, an OO development process has three stages:
  1. OO analysis
  2. OO design
  3. OO implementation
- To illustrate this process, we are going to use an Order Processing System (OPS)

OO Analysis and Design

- (RI) Each customer has an online shopping cart, which allows addition and removal of items.
- (R2) A customer is able to check out the shopping cart using credit card. The transaction is approved only when the financial department has verified the validity and the remaining credit line of the credit card.
- (RJ) Before the transaction is completed, a customer should be able to learn about the estimated arrival date, which is determined by the order processing time of the shipping department.
- (R4) A customer can choose to cancel the transaction by clearing all items of the cart.
- (R5) OPS must be available as a Web accessible system. Customers can use popular Internet browsers to interact with OPS.

OO Analysis and Design
(OO Analysis)

- The purpose of OO analysis is to understand the domain and requirements of the system

OO Analysis and Design
(OO Analysis)

- The purpose of OO analysis is to understand the domain and requirements of the system
- The outcome of the analysis stage is a requirement specification

OO Analysis and Design
(OO Analysis)

- The purpose of OO analysis is to understand the domain and requirements of the system
- The outcome of the analysis stage is a requirement specification
- OO analysis mainly relies on two UML tools:

OO Analysis and Design
(OO Analysis)

- The purpose of OO analysis is to understand the domain and requirements of the system
- The outcome of the analysis stage is a requirement specification
- OO analysis mainly relies on two UML tools:
  - UML Use Case Diagram ⟸ Reqts

OO Analysis and Design
(OO Analysis)

- The purpose of OO analysis is to understand the domain and requirements of the system
- The outcome of the analysis stage is a requirement specification
- OO analysis mainly relies on two UML tools:
  - UML Use Case Diagram ⟸ Reqts
  - Analysis Class Diagram ⟸ Design

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram

- Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram

- Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.
- Session: When a customer logs on using correct user name and password, the customer starts a session. During the session, the system maintains a shopping cart for the user, which records the list of desired items that the customer wants to purchase later. The session is terminated either because the user checks out the shopping cart, logs out of the system, or times out. A session consists of a list of operations.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Operation: The operation use case is an abstract use case. It is extended by login, logout, add item, remove item, clear cart, and check out.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Operation: The operation use case is an abstract use case. It is extended by login, logout, add item, remove item, clear cart, and check out.
- Log in: It is always the first operation of any session. When the supplied user name and password are correct, the session starts. A maximum number of 3 trials are allowed for each user. When a user fails more than 3 times, the account is locked.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Operation: The operation use case is an abstract use case. It is extended by login, logout, add item, remove item, clear cart, and check out.
- Log in: It is always the first operation of any session. When the supplied user name and password are correct, the session starts. A maximum number of 3 trials are allowed for each user. When a user fails more than 3 times, the account is locked.
- Log out: When the customer has not checked out all the items of the shopping cart, the shopping cart is saved in the database so that at the beginning of the next logs on, the shopping cart is loaded with the items of the last session.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Add item: Add the item into the shopping cart. Before the item is added, the catalog and inventory database should be checked to verify availability of the item, and the cost and price information of the item.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Add item: Add the item into the shopping cart. Before the item is added, the catalog and inventory database should be checked to verify availability of the item, and the cost and price information of the item.
- Remove item: Remove the item from the shopping cart, given that the item is already in it. The item is put back into the inventory database.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Add item: Add the item into the shopping cart. Before the item is added, the catalog and inventory database should be checked to verify availability of the item, and the cost and price information of the item.
- Remove item: Remove the item from the shopping cart, given that the item is already in it. The item is put back into the inventory database.
- Clear shopping cart: Remove all items from the chart.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Add item: Add the item into the shopping cart. Before the item is added, the catalog and inventory database should be checked to verify availability of the item, and the cost and price information of the item.
- Remove item: Remove the item from the shopping cart, given that the item is already in it. The item is put back into the inventory database.
- Clear shopping cart: Remove all items from the chart.
- Check out: Ask the user to input credit card information, send the information to financial department for verification. If the transaction is approved, the credit card is charged; if not, prompts the user that the check out process has failed, and the items still remain in the cart. When trans. is successful, instructions are sent to shipping department.

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

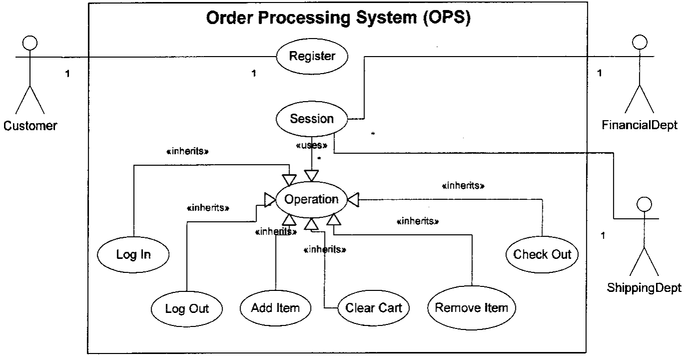


Figure: Use case diagram of the OPS system

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Usually in an 00 design, each use case is monitored by a controller class, especially when the use case has a long duration flow of events

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Usually in an OO design, each use case is monitored by a controller class, especially when the use case has a long duration flow of events

- Sometimes, a use case can be modeled as an operation, if its logic is not complex

OO Analysis and Design
(OO Analysis)
STEP 1: Use Case Diagram (Continued)

- Usually in an OO design, each use case is monitored by a controller class, especially when the use case has a long duration flow of events
- Sometimes, a use case can be modeled as an operation, if its logic is not complex
- How to develop an analysis class diagram based on the information we have obtained during the requirement analysis?

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- An analysis class diagram describes the key classes of the system and their interrelationship

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- An analysis class diagram describes the key classes of the system and their interrelationship

- The analysis class diagram is NOT the design of the system

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- An analysis class diagram describes the key classes of the system and their interrelationship

- The analysis class diagram is NOT the design of the system

- An analysis class diagram describes the functionality of the system

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- An analysis class diagram describes the key classes of the system and their interrelationship
- The analysis class diagram is NOT the design of the system
- An analysis class diagram describes the functionality of the system
- It is an abstract model of the system by abstracting out the design details (overall architecture)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- An analysis class diagram describes the key classes of the system and their interrelationship

- The analysis class diagram is NOT the design of the system

- An analysis class diagram describes the functionality of the system

- It is an abstract model of the system by abstracting out the design details (overall architecture)

- The class diagram will be further refined and the relationship among the classes might be changed (add Nonfunctional Rqts)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
  - Serve as the interface between the internal of the system and the outside world

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
  - Serve as the interface between the internal of the system and the outside world
  - In the form of graphic user interface classes and wrapper classes of other systems

OO Analysis and Design (OO Analysis) STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
    - Serve as the interface between the internal of the system and the outside world
    - In the form of graphic user interface classes and wrapper classes of other systems
- Entity classes:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
  - Serve as the interface between the internal of the system and the outside world
  - In the form of graphic user interface classes and wrapper classes of other systems
- Entity classes:
  - Represent the information stored and exchanged among components of the system

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
  - Serve as the interface between the internal of the system and the outside world
  - In the form of graphic user interface classes and wrapper classes of other systems
- Entity classes:
  - Represent the information stored and exchanged among components of the system
- Controller classes:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
  - Serve as the interface between the internal of the system and the outside world
  - In the form of graphic user interface classes and wrapper classes of other systems
- Entity classes:
  - Represent the information stored and exchanged among components of the system
- Controller classes:
  - Coordinates the interactions that are required to accomplish the use case

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

There are three types of classes:

- Boundary classes:
    - Serve as the interface between the internal of the system and the outside world
    - In the form of graphic user interface classes and wrapper classes of other systems
- Entity classes:
    - Represent the information stored and exchanged among components of the system
- Controller classes:
    - Coordinates the interactions that are required to accomplish the use case
    - Are event handlers

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

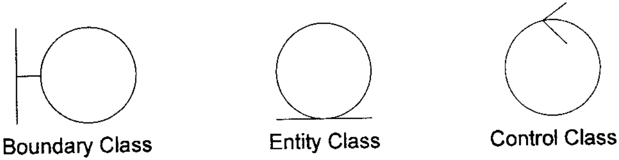


Figure: Three types of classes in an analysis class diagram

STEP 2: Develop Analysis Class Diagram

- How to identify classes which constitute the system first?

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- How to identify classes which constitute the system first?

- Extract nouns and verbs in the use case descriptions

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

STEP 2: Develop Analysis Class Diagram

- How to identify classes which constitute the system first?
- Extract nouns and verbs in the use case descriptions
- Nouns are candidates of classes

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- How to identify classes which constitute the system first?
- Extract nouns and verbs in the use case descriptions
- Nouns are candidates of classes
- Verbs are candidates of their functions

  Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

STEP 2: Develop Analysis Class Diagram

Should every noun becomes a class of the system????

- customer: No (outside of the system)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

Should every noun becomes a class of the system????

- customer: No (outside of the system)
- registration page: Yes (boundary class)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

Should every noun becomes a class of the system????

- customer: No (outside of the system)
- registration page: Yes (boundary class)
- user name: No (an attribute of another class: CustomerIdentit)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

Should every noun becomes a class of the system????

- customer: No (outside of the system)
- registration page: Yes (boundary class)
- user name: No (an attribute of another class: CustomerIdentit)
- password: No (an attribute of another class: CustomerIdentit)

OO Analysis and Design

(OO Analysis)

STEP 2: Develop Analysis Class Diagram

Should every noun becomes a class of the system????

- customer: No (outside of the system)
- registration page: Yes (boundary class)
- user name: No (an attribute of another class: CustomerIdentit)
- password: No (an attribute of another class: CustomerIdentit)
- username/password pair: Yes (should be modeled as CustomerIdentity class)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase ”system reports error” is not accurate enough
  - can be rephrased as ”errors are reported in a RegistrationErrorPage”

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough
  - can be rephrased as "errors are reported in a RegistrationErrorPage"
  - a RegistrationErrorPage should be created as a boundary class

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
    - the phrase "system reports error" is not accurate enough
    - can be rephrased as "errors are reported in a RegistrationErrorPage"
    - a RegistrationErrorPage should be created as a boundary class

- error: No

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough
  - can be rephrased as "errors are reported in a RegistrationErrorPage"
  - a RegistrationErrorPage should be created as a boundary class
- error: No
- success: ⟹ RegistrationSuccessPage (boundary class)

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough
  - can be rephrased as "errors are reported in a RegistrationErrorPage"
  - a RegistrationErrorPage should be created as a boundary class
- error: No
- success: ⟹ RegistrationSuccessPage (boundary class)
- registration:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough
  - can be rephrased as "errors are reported in a RegistrationErrorPage"
  - a RegistrationErrorPage should be created as a boundary class
- error: No
- success: $\Longrightarrow$ RegistrationSuccessPage (boundary class)
- registration:
  - refers to the registration process

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- system:
  - the phrase "system reports error" is not accurate enough
  - can be rephrased as "errors are reported in a RegistrationErrorPage"
  - a RegistrationErrorPage should be created as a boundary class

- error: No

- success: $\Longrightarrow$ RegistrationSuccessPage (boundary class)

- registration:
  - refers to the registration process
  - leads to the creation of controller class named RegistrationController for this use case

OO Analysis and Design (OO Analysis) STEP 2: Develop Analysis Class Diagram

- real name: No

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- real name: No
- billing address: No

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- real name: No
- billing address: No
- database system:

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- real name: No
- billing address: No
- database system:
  - we can infer that there are catalog and inventory database systems out of the scope of OPS

STEP 2: Develop Analysis Class Diagram

- real name: No
- billing address: No
- database system:
  - we can infer that there are catalog and inventory database systems out of the scope of OPS
  - we need to maintain a database within OPS for storing all customer information

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- real name: No
- billing address: No
- database system:
  - we can infer that there are catalog and inventory database systems out of the scope of OPS
  - we need to maintain a database within OPS for storing all customer information
  - a wrapper class (i.e., boundary class) called CustomerInfoDB should be created to interact with the database

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

We obtain:

- RegistrationPage: a boundary class for registration

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

We obtain:

- RegistrationPage: a boundary class for registration
- RegistrationErrorPage: a boundary class for reporting registration error

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

We obtain:

- RegistrationPage: a boundary class for registration
- RegistrationErrorPage: a boundary class for reporting registration error
- RegistrationSuccessPage: a boundary class for concluding the registration process

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

We obtain:

- RegistrationPage: a boundary class for registration
- RegistrationErrorPage: a boundary class for reporting registration error
- RegistrationSuccessPage: a boundary class for concluding the registration process
- Customer/identity: an entity class which records the user name and password of a customer

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- CustomerInfomation: an entity class which records the information of a customer, e.g., his/her CustomerIdentity, real name, and billing address

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- CustomerInfomation: an entity class which records the information of a customer, e.g., his/her CustomerIdentity, real name, and billing address

- CustomerInfoDB: a boundary class which manipulates the customer information database

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

- CustomerInfomation: an entity class which records the information of a customer, e.g., his/her CustomerIdentity, real name, and billing address
- CustomerInfoDB: a boundary class which manipulates the customer information database
- RegistrationController: a controller class which coordinates the registration process. The class is responsible for, e.g., , redirecting from the RegistrationPage to RegistrationSuccessPage if the registration is completed successfully

OO Analysis and Design
(OO Analysis)
STEP 2: Develop Analysis Class Diagram

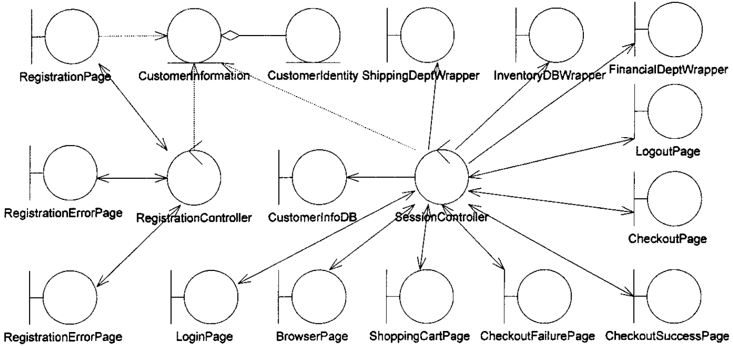


Figure: Analysis class diagram of OPS

OO Analysis and Design (OO Design)

- The goal of a design process is to develop the overall structure of the system

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system

- In a OO design, a system is decomposed into logical components in the form of classes

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system
- In a OO design, a system is decomposed into logical components in the form of classes
- For each class, its interface is articulated in the design stage

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system
- In a OO design, a system is decomposed into logical components in the form of classes
- For each class, its interface is articulated in the design stage
- In the implementation stage, the design of each class is translated into code using an OO language

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system
- In a OO design, a system is decomposed into logical components in the form of classes
- For each class, its interface is articulated in the design stage
- In the implementation stage, the design of each class is translated into code using an OO language
- OO design can be split into two stages:

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system
- In a OO design, a system is decomposed into logical components in the form of classes
- For each class, its interface is articulated in the design stage
- In the implementation stage, the design of each class is translated into code using an OO language
- OO design can be split into two stages:
  - high level design (class diagram, interaction diagrams)

OO Analysis and Design
(OO Design)

- The goal of a design process is to develop the overall structure of the system
- In a OO design, a system is decomposed into logical components in the form of classes
- For each class, its interface is articulated in the design stage
- In the implementation stage, the design of each class is translated into code using an OO language
- OO design can be split into two stages:
  - high level design (class diagram, interaction diagrams)
  - low level design (attributes and operations of each class, SM diagram/Formal spec. for each class)

OO Analysis and Design (OO Design)

STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system

- Classes included in the diagram may be only a subset of all the classes that are needed

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system
- Classes included in the diagram may be only a subset of all the classes that are needed
- CRC (Class-Responsibility-Collaborator) Card Modeling is a simple analysis technique for

STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system
- Classes included in the diagram may be only a subset of all the classes that are needed
- CRC (Class-Responsibility-Collaborator) Card Modeling is a simple analysis technique for
    - identifying classes to build a system

STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system
- Classes included in the diagram may be only a subset of all the classes that are needed
- CRC (Class-Responsibility-Collaborator) Card Modeling is a simple analysis technique for
  - identifying classes to build a system
  - assigning responsibilities to classes

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

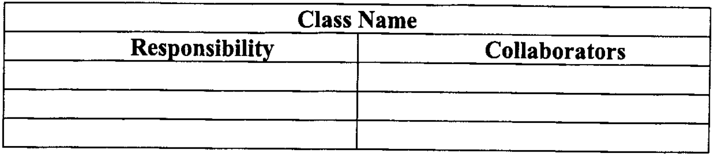
- Analysis class diagram presents the rough logical structure of the system
- Classes included in the diagram may be only a subset of all the classes that are needed
- CRC (Class-Responsibility-Collaborator) Card Modeling is a simple analysis technique for
    - identifying classes to build a system
    - assigning responsibilities to classes
- CRC card has been widely used for OO analysis in practice

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- Analysis class diagram presents the rough logical structure of the system
- Classes included in the diagram may be only a subset of all the classes that are needed
- CRC (Class-Responsibility-Collaborator) Card Modeling is a simple analysis technique for
    - identifying classes to build a system
    - assigning responsibilities to classes
- CRC card has been widely used for OO analysis in practice
- CRC card modeling is a team work (will explain how to ply)

OO Analysis and Design (OO Design)

STEP 1: Identify all Classes



Class Name	
Responsibility	Collaborators

Figure: CRC card template

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- A class may already exist in the analysis class diagram

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- A class may already exist in the analysis class diagram
- Or, it is included in the design due to the need of another class

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- A class may already exist in the analysis class diagram
- Or, it is included in the design due to the need of another class
- A responsibility is a task that must be performed by the class, or the knowledge that is hold by the class

STEP 1: Identify all Classes

- A class may already exist in the analysis class diagram
- Or, it is included in the design due to the need of another class
- A responsibility is a task that must be performed by the class, or the knowledge that is hold by the class
- A collaborator is a class that is involved in accomplishing a responsibility

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:
    - domain user

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:
  - domain user
  - OO Design Analyst

OO Analysis and Design (OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:
  - domain user
  - OO Design Analyst
  - facilitator

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:
    - domain user
    - OO Design Analyst
    - facilitator

- The initial cards may come from the analysis class diagram, or existing design classes

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

How to play CRC cards?

- A CRC team generally consists of three types of participants:
    - domain user
    - OO Design Analyst
    - facilitator
- The initial cards may come from the analysis class diagram, or existing design classes
- Then, each OO Design Analyst is assigned one or more cards and then plays the role of the cards

OO Analysis and Design (OO Design)

STEP 1: Identify all Classes

- The team examines the set of use cases one by one, simulates the interaction and flow of events that are used to accomplish the use case

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- The team examines the set of use cases one by one, simulates the interaction and flow of events that are used to accomplish the use case

- When a class is involved in the execution of the interaction,

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- The team examines the set of use cases one by one, simulates the interaction and flow of events that are used to accomplish the use case
- When a class is involved in the execution of the interaction,
  - the responsibility is recorded for that class

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

- The team examines the set of use cases one by one, simulates the interaction and flow of events that are used to accomplish the use case

- When a class is involved in the execution of the interaction,
  - the responsibility is recorded for that class
  - the class role player has to enumerate the actions performed by the class and the other classes (collaborators) that are involved

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

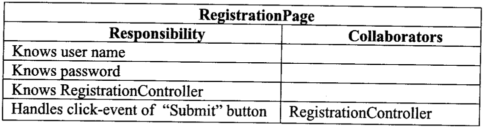
- The team examines the set of use cases one by one, simulates the interaction and flow of events that are used to accomplish the use case

- When a class is involved in the execution of the interaction,
  - the responsibility is recorded for that class
  - the class role player has to enumerate the actions performed by the class and the other classes (collaborators) that are involved
  - the session continues for all use cases and all details of each responsibility of each class are clear

OO Analysis and Design (OO Design)

STEP 1: Identify all Classes

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.

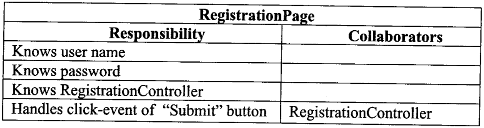


RegistrationPage	
Responsibility	**Collaborators**
Knows user name	
Knows password	
Knows RegistrationController	
Handles click-event of "Submit" button	RegistrationController

Figure: CRC card of RegistrationPage

OO Analysis and Design
(OO Design)
STEP 1: Identify all Classes

Register: A customer visits the registration page, and specifies the desired user name and password. If there is already such a username/password pair, the system reports error; otherwise, it reports success. When the registration is completed, the user name, password, real name, and the billing address are stored in a database system.



RegistrationPage	
Responsibility	**Collaborators**
Knows user name	
Knows password	
Knows RegistrationController	
Handles click-event of "Submit" button	RegistrationController

Figure: CRC card of RegistrationPage

Read Chap. 4 for the rest of the example

OO Analysis and Design (OO Design)
STEP 2: Construct Interaction Diagram

- Based on the use case and the CRC cards generated, we construct the interaction diagram for each use case

OO Analysis and Design
(OO Design)
STEP 2: Construct Interaction Diagram

- Based on the use case and the CRC cards generated, we construct the interaction diagram for each use case

- An interaction diagram describes how the objects of the system interact with each other to accomplish the use case

OO Analysis and Design
(OO Design)
STEP 2: Construct Interaction Diagram

- Based on the use case and the CRC cards generated, we construct the interaction diagram for each use case
- An interaction diagram describes how the objects of the system interact with each other to accomplish the use case
- Use sequence diagram $\oplus$ communication diagram

OO Analysis and Design
(OO Design)
STEP 2: Construct Interaction Diagram

- Based on the use case and the CRC cards generated, we construct the interaction diagram for each use case
- An interaction diagram describes how the objects of the system interact with each other to accomplish the use case
- Use sequence diagram $\oplus$ communication diagram
- In a sequence diagram, the boxes on the top of the diagram denote objects, classes, or actors

STEP 2: Construct Interaction Diagram

- Based on the use case and the CRC cards generated, we construct the interaction diagram for each use case
- An interaction diagram describes how the objects of the system interact with each other to accomplish the use case
- Use sequence diagram $\oplus$ communication diagram
- In a sequence diagram, the boxes on the top of the diagram denote objects, classes, or actors
- "successPage:RegistrationSuccessPage" $\Longrightarrow$ successPage is the object name, and RegistrationSuccessPage is the class name

OO Analysis and Design
(OO Design)
STEP 2: Construct Interaction Diagram

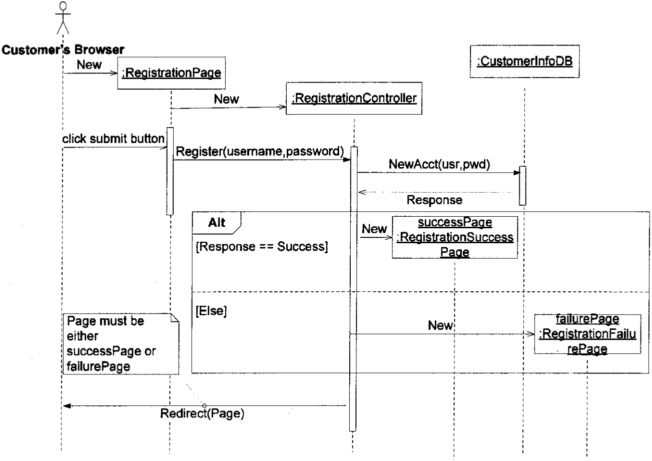


Figure: Sequence diagram for the registration use case

OO Analysis and Design

(OO Design)

STEP 3: Build State Machine Diagram

- Dynamic behaviors of some objects, especially those controller objects, may be very complex

OO Analysis and Design
(OO Design)
STEP 3: Build State Machine Diagram

- Dynamic behaviors of some objects, especially those controller objects, may be very complex

- It is beneficial to describe the whole life cycle of a complex object using state machine diagram

STEP 3: Build State Machine Diagram

- Dynamic behaviors of some objects, especially those controller objects, may be very complex

- It is beneficial to describe the whole life cycle of a complex object using state machine diagram

- How to build a state diagram?

OO Analysis and Design
(OO Design)
STEP 3: Build State Machine Diagram

- Dynamic behaviors of some objects, especially those controller objects, may be very complex

- It is beneficial to describe the whole life cycle of a complex object using state machine diagram

- How to build a state diagram?
  - When the class is responsible for a single use case, we simply look into the use case + corresponding interaction diagram, identify the different stages that the control class goes through

OO Analysis and Design
(OO Design)
STEP 3: Build State Machine Diagram

- Dynamic behaviors of some objects, especially those controller objects, may be very complex
- It is beneficial to describe the whole life cycle of a complex object using state machine diagram
- How to build a state diagram?
  - When the class is responsible for a single use case, we simply look into the use case + corresponding interaction diagram, identify the different stages that the control class goes through
  - When the class is involved in multiple use cases, we have to combine all the information to identify the states

OO Analysis and Design (OO Design)

STEP 3: Build State Machine Diagram

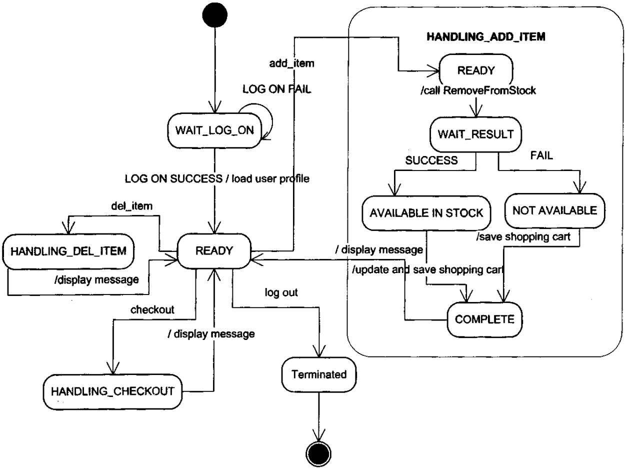


Figure: State machine diagram of SessionController class

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The high level design is used to identify the classes and objects that are used to form the system

OO Analysis and Design (OO Design)
STEP 4: Detailed Design

- The high level design is used to identify the classes and objects that are used to form the system
- The detailed level design has to identify the interface and implementation details of each class

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The high level design is used to identify the classes and objects that are used to form the system
- The detailed level design has to identify the interface and implementation details of each class
- The interface of a class is the public interconnection border of the class that is accessible by other components of the system

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The high level design is used to identify the classes and objects that are used to form the system
- The detailed level design has to identify the interface and implementation details of each class
- The interface of a class is the public interconnection border of the class that is accessible by other components of the system
- The implementation is how to realize the behaviors as specified by the interface of a class

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)

- When formal methods are used, for each method give:

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)

- When formal methods are used, for each method give:
  - pre-condition

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)

- When formal methods are used, for each method give:
  - pre-condition

  - postcondition

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)
- When formal methods are used, for each method give:
    - pre-condition
    - postcondition
    - invariant

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)
- When formal methods are used, for each method give:
  - pre-condition
  - postcondition
  - invariant
  - (see your notes from SFWR ENG 2A04)

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The outcome of the DD process is a detailed class design diagram (list of attributes and operations)
- When formal methods are used, for each method give:
    - pre-condition
    - postcondition
    - invariant
    - (see your notes from SFWR ENG 2A04)
- From the DD, the implementation in code should be straightforward

OO Analysis and Design (OO Design)

STEP 4: Detailed Design

- The first step in the detailed design is to identify the set of public attributes and operations of a class

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The first step in the detailed design is to identify the set of public attributes and operations of a class

- An interface of a class: all attributes and operations are "public"

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The first step in the detailed design is to identify the set of public attributes and operations of a class

- An interface of a class: all attributes and operations are "public"

- In UML, these public members are preceded by the notation "+"

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The first step in the detailed design is to identify the set of public attributes and operations of a class
- An interface of a class: all attributes and operations are "public"
- In UML, these public members are preceded by the notation "+"
- The designers and programmers of other components will rely on the public interface of a class

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- The first step in the detailed design is to identify the set of public attributes and operations of a class
- An interface of a class: all attributes and operations are "public"
- In UML, these public members are preceded by the notation "+"
- The designers and programmers of other components will rely on the public interface of a class
- The class can be split into more classes in the detailed design, if it is too complex

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- In practice, a class can declare public attributes

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- In practice, a class can declare public attributes
- A class provides four categories of operations:

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- In practice, a class can declare public attributes
- A class provides four categories of operations:
    - constructor (initialize the data members of an object)

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- In practice, a class can declare public attributes
- A class provides four categories of operations:
  - constructor (initialize the data members of an object)
  - destructor (clean-off memory and free system resources)

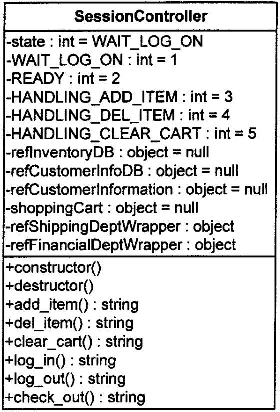
OO Analysis and Design (OO Design) STEP 4: Detailed Design

- In practice, a class can declare public attributes
- A class provides four categories of operations:
  - constructor (initialize the data members of an object)
  - destructor (clean-off memory and free system resources)
  - accessor (retrieves the information from the object)

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design

- In practice, a class can declare public attributes
- A class provides four categories of operations:
  - constructor (initialize the data members of an object)
  - destructor (clean-off memory and free system resources)
  - accessor (retrieves the information from the object)
  - mutator (changes the state of the object)

OO Analysis and Design
(OO Design)
STEP 4: Detailed Design



SessionController

```
-state : int = WAIT_LOG_ON
-WAIT_LOG_ON : int = 1
-READY : int = 2
-HANDLING_ADD_ITEM : int = 3
-HANDLING_DEL_ITEM : int = 4
-HANDLING_CLEAR_CART : int = 5
-refInventoryDB : object = null
-refCustomerInfoDB : object = null
-refCustomerInformation : object = null
-shoppingCart : object = null
-refShippingDeptWrapper : object
-refFinancialDeptWrapper : object
```

```
+constructor()
+destructor()
+add_item() : string
+del_item() : string
+clear_cart() : string
+log_in() : string
+log_out() : string
+check_out() : string
```

Figure: Detailed design of Session Controller class