

Software Design and Construction 1

SOFT2201 / COMP9201

Assignment Project Exam Help

UML Modeling

<https://powcoder.com>

Dr. Xi Wu

Add WeChat powcoder



School of Computer Science

Copyright warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

Assignment Project Exam Help

WARNING

This material has been reproduced and communicated to
you by or on behalf of the University of Sydney
pursuant to Part VB of the Copyright Act 1968 (the
Act).

Add WeChat powcoder

The material in this communication may be subject
to copyright under the Act. Any further copying or
communication of this material by you may be the
subject of copyright protection under
the Act.

Do not remove this notice.

Agenda

- UML Modeling
 - UML Use Case Diagrams
 - UML Class Diagrams
 - UML Interaction Diagrams
- Assignment Project Exam Help
<https://powcoder.com>
- Case Study: Next Gen Point-of-Sale (POS) System
Add WeChat powcoder
 - Brief Introduction
 - Self learning details

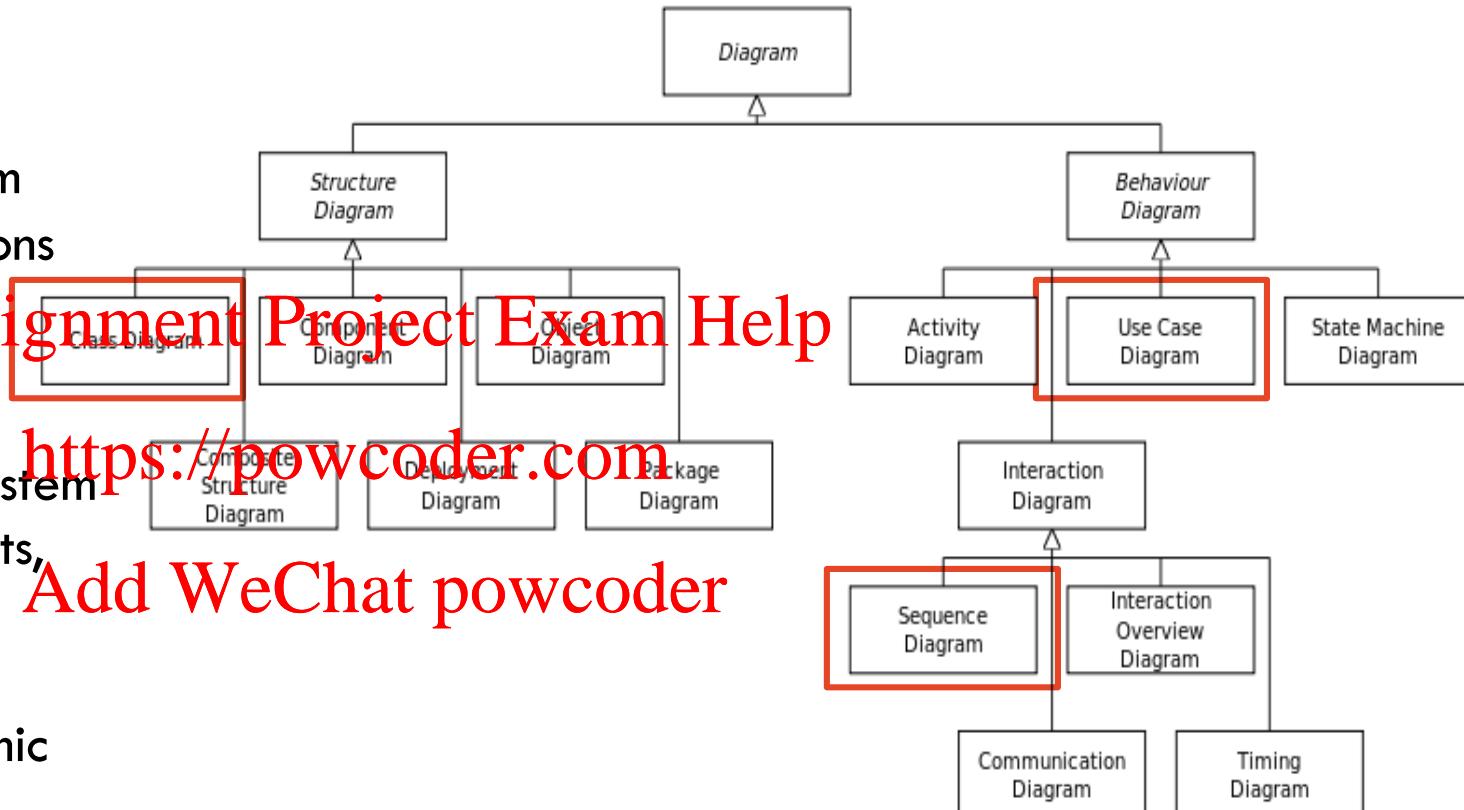
Software Modelling

- The process of developing conceptual models of a system at different levels of **abstraction**
 - Fulfil the defined system requirements
 - Focus on important system details to deal with complexity
- Using graphical notations (e.g., UML) to capture different views or perspectives
 - There are three ways to apply UML
 - Blueprint: detailed design diagrams for:
 - Reverse engineering to visualize and understand existing code
 - Forward engineering (code generation)

UML Diagrams

Structural (static) View

- Static structure of the system (objects, attributes, operations and relationships)



Behavioural (dynamic) View

- Dynamic behavior of the system (collaboration among objects, and changes to the internal states of objects)
- Interaction (subset of dynamic view) - emphasizes flow of control and data

https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/UML

Use Case Diagrams

Assignment Project Exam Help

1. Why we need Use Case Diagrams?

2. What is Use Case?

3. How can we draw Use Case Diagrams?

<https://powcoder.com>

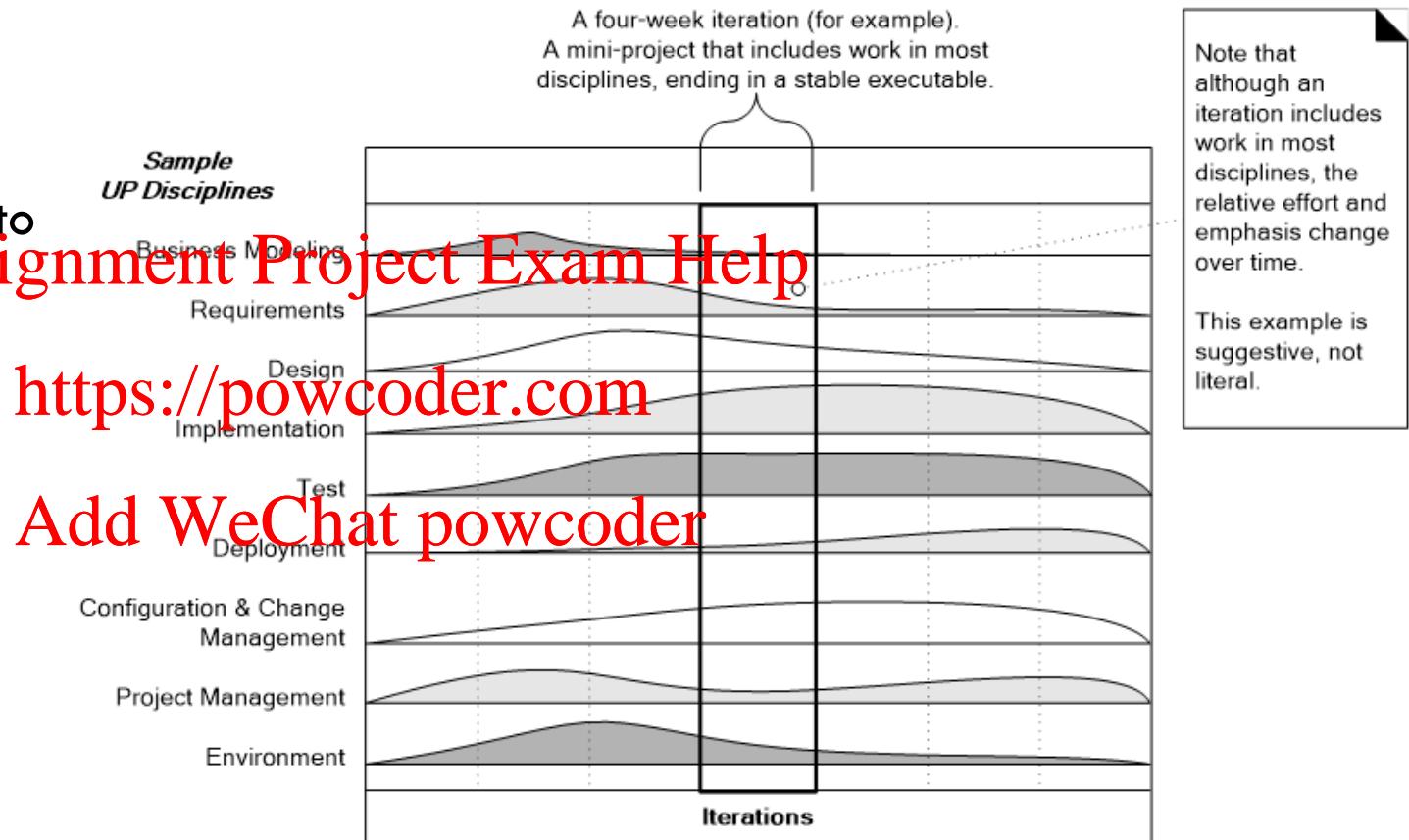
Add WeChat powcoder

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



Rational Unified Process (UP)

- Software development process utilizing iterative and risk-driven approach to develop OO software systems
- Iterative incremental development
- Iterative evolutionary development

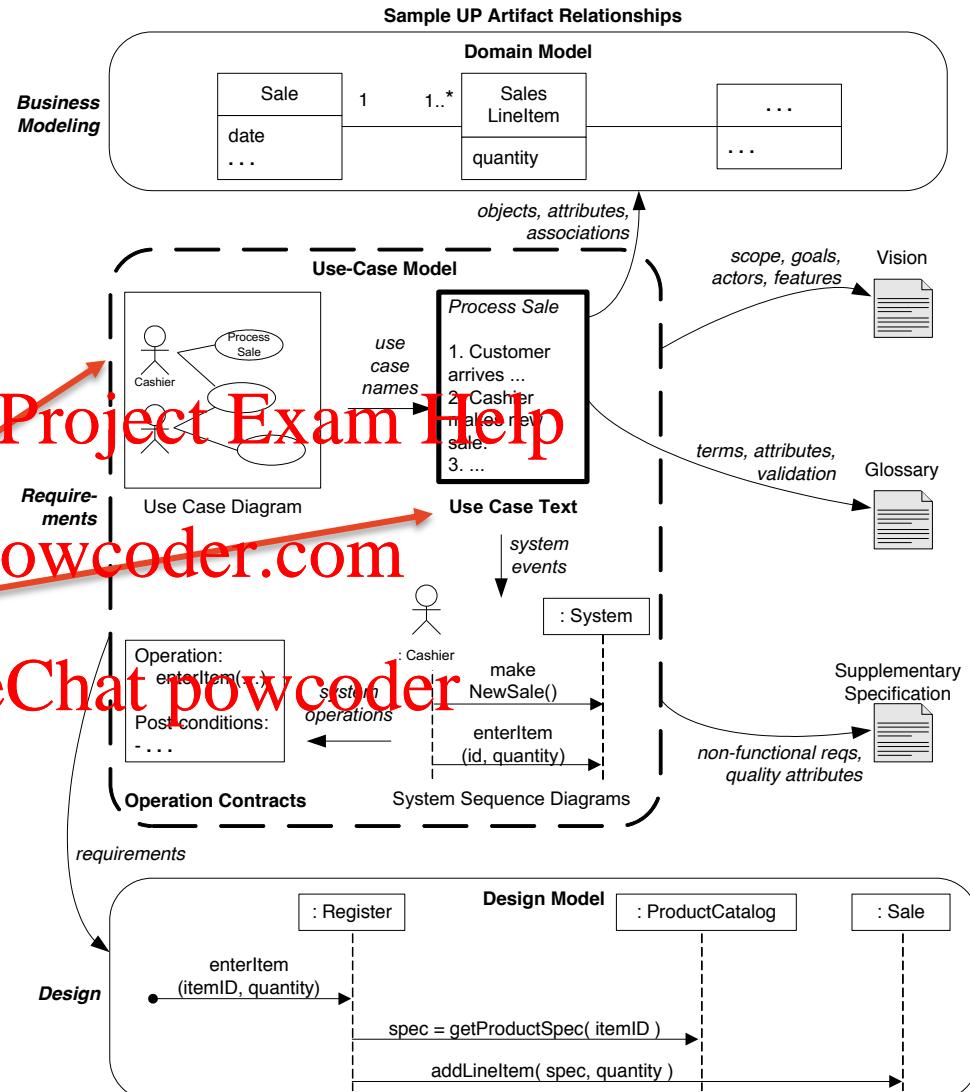


Business Modelling, Requirements and Designs in RUP/UP*

Use Cases and Use Case Diagrams

*RUP/UP: Rational Unified Process,
Lecture 1 pp. 59--64

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



Use Cases

- **Use case:** “specifies a set of behaviors performed by a system, which yields an observable result that is of value for Actors or other stakeholders of the system”*
 - It capture what a system supposed to do (system’s requirements)
 - Text documents
 - Primary part of the use case model
- **Scenario (use case instance):** specific sequence of action and interactions between actors and the system
 - One particular story of using a system (e.g., successfully purchasing items with cash)
- **Use case model:** set of all written use cases (a model of the system functionality and environment)
 - Defined within the Requirements discipline in the UP
 - Not the only requirement artefact – business rules, supplementary specifications, etc

* OMG Unified Modeling Language, version 2.5.1, Dec. 2017 <https://www.omg.org/spec/UML/2.5.1>

Use Cases – Common Formats

- **Brief:** one-paragraph summary, usually of the main success scenario
 - During early requirements analysis to get quick subject and scope
- **Casual:** Informal paragraph format; multiple paragraphs that cover various scenarios
 - During early requirements analysis to get quick subject and scope
- **Full-dressed:** all steps and variations are written in detail and there are supporting sections; e.g., pre-conditions, success guarantee
 - After many use cases have been identified and written in brief format

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

* OMG Unified Modeling Language, version 2.5.1, Dec. 2017 <https://www.omg.org/spec/UML/2.5.1>

Use Cases – Full-dressed Template

Use case section	Comment
Use case name	Start with a verb
Scope	The system under design
Level	“User-goal” or sub-function
Primary actor	Calls on the system to deliver its services
Stakeholders and interests	Who cares about this UC and what do they earn?
Preconditions	What must be true on start
Success guarantee	What must be true on successful completion
Main success scenario	Typical unconditional happy path scenario
Extensions	Alternate scenarios of success or failure
Special requirements	Related non-functional requirement
.....	

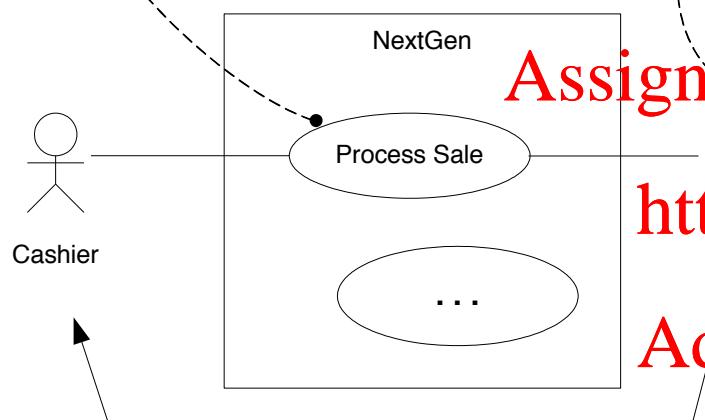
Use Case Diagrams

- UML graphical notations that help to capture use cases (system's boundaries and interactions and relationships)
 - **Subject:** system under consideration to which the use case applies
 - **Actor:** role that interacts with the subject/system (e.g., end user, customer, supplier, another system)
 - **Use case:** describes functionality of the system
 - **Association:** relationship between an actor and a use case (an actor can use certain functionality of the system)
 - «include» indicates the behavior of the included use case is included in the behavior of the including use case

* OMG Unified Modeling Language, version 2.5.1, Dec. 2017 <https://www.omg.org/spec/UML/2.5.1>

Use Case Diagram – UML Notations

For a use case context diagram, limit the use cases to user-goal level use cases.



Show computer system actors with an alternate notation to human actors.

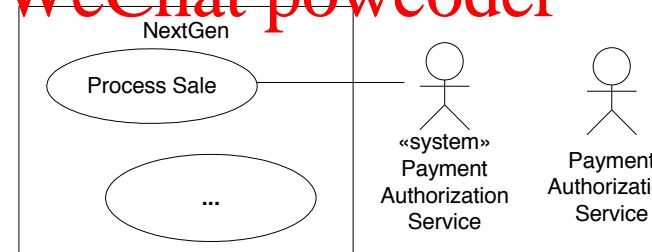
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

primary actors on the left

supporting actors on the right



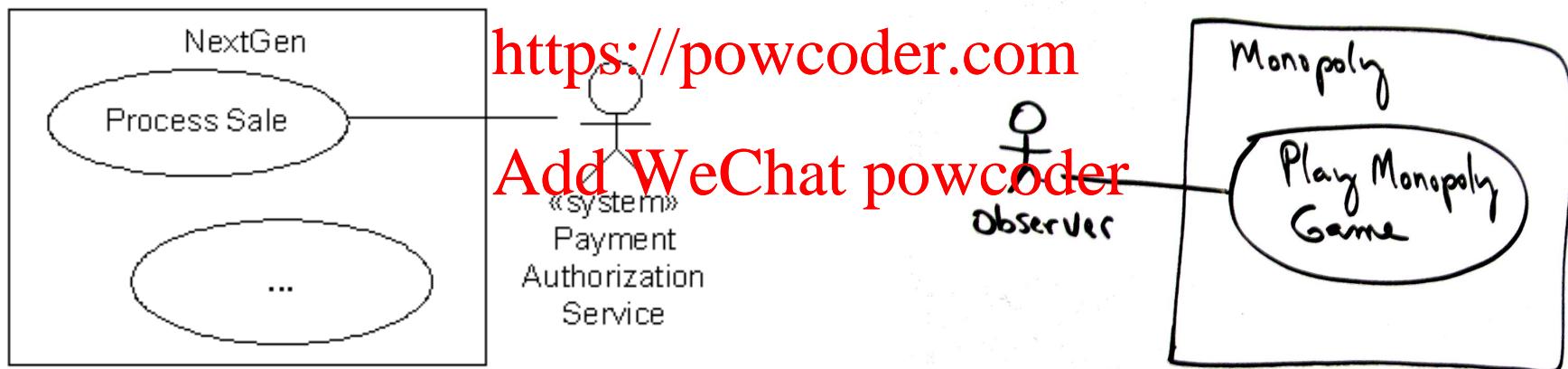
Some UML alternatives to illustrate external actors that are other computer systems.

The class box style can be used for any actor, computer or human. Using it for computer actors provides visual distinction.

Use Case Diagrams – Tools

- There are many tools to aid drawing UML diagrams
 - Tools are means to make your life easier
 - You can also sketch diagrams using pen-and-paper or white board

Assignment Project Exam Help



https://www.lucidchart.com/pages/examples/uml_diagram_tool

Class Diagrams

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Structural Diagrams

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



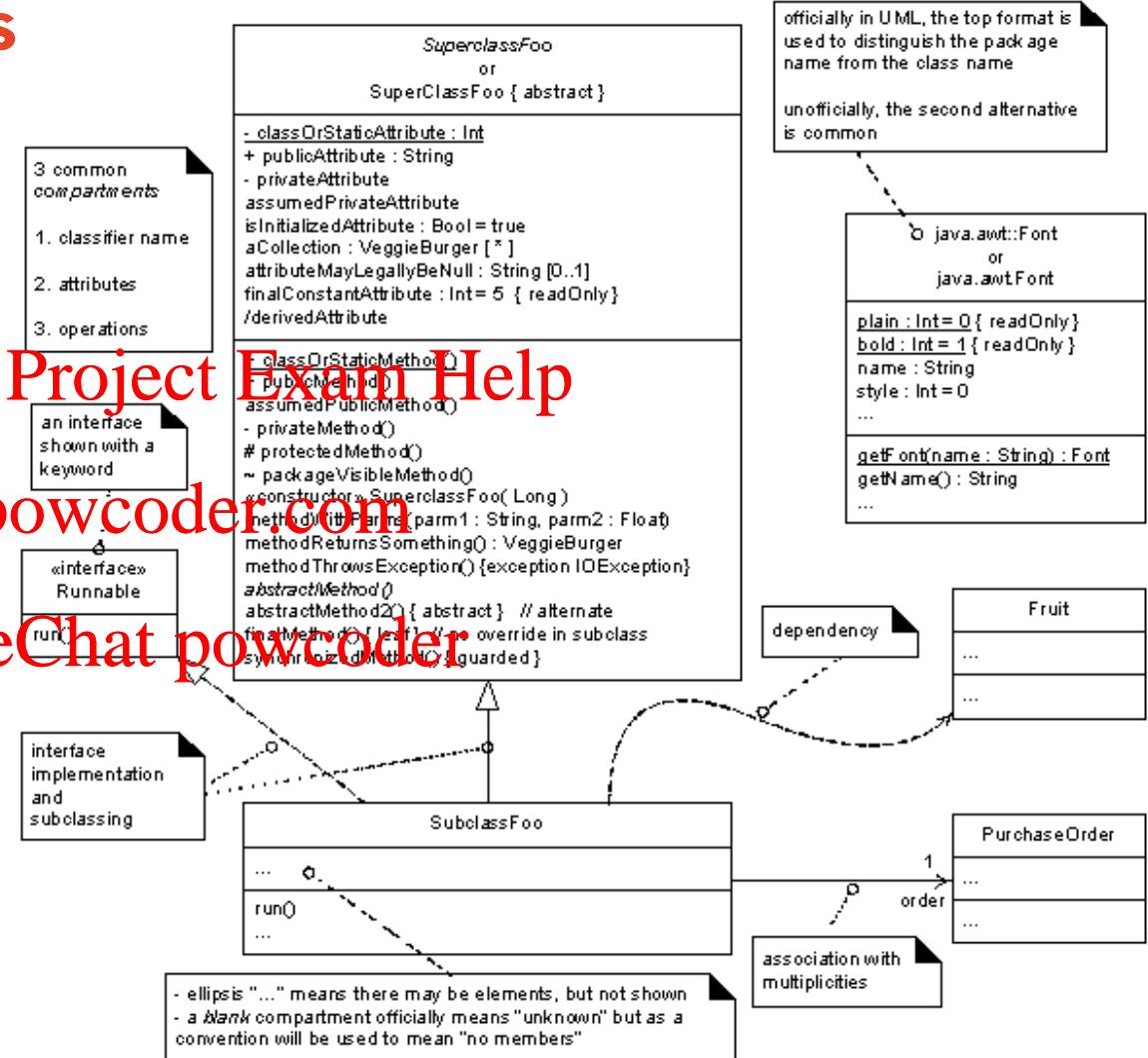
Class Diagram – Notations

- Common compartments: classifier name, attributes and operations
 - Package name
 - <<interface>>
- Class hierarchy – inheritance
- Dependency
- Association and multiplicity
- Optional and default elements

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Class Diagram – Perspectives

- **Conceptual:** describes key concepts in the problem domain. Use in business modeling for OO analysis.

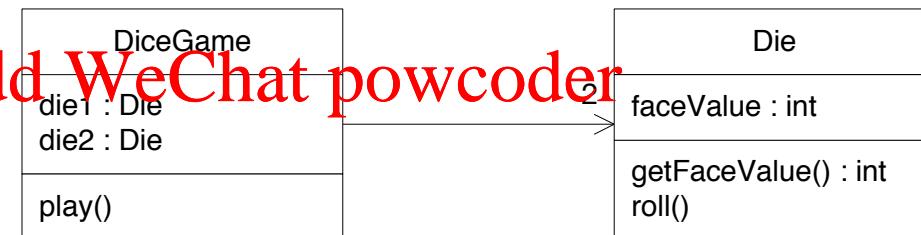


Conceptual Perspective
(domain model)

Raw UML class diagram notation used to visualize real-world concepts.

- **Specification:** describes software components with specification and interfaces
- **Implementation:** describes software implementation in a particular programming language (e.g., Java)

<https://powcoder.com>



Specification or
Implementation
Perspective
(design class diagram)

Raw UML class diagram notation used to visualize software elements.

Class Diagrams – UML Attributes

Attribute Text

Visibility : type {property string}

Visibility + (public), - (private)

Attributes are assumed private if no visibility sign shown

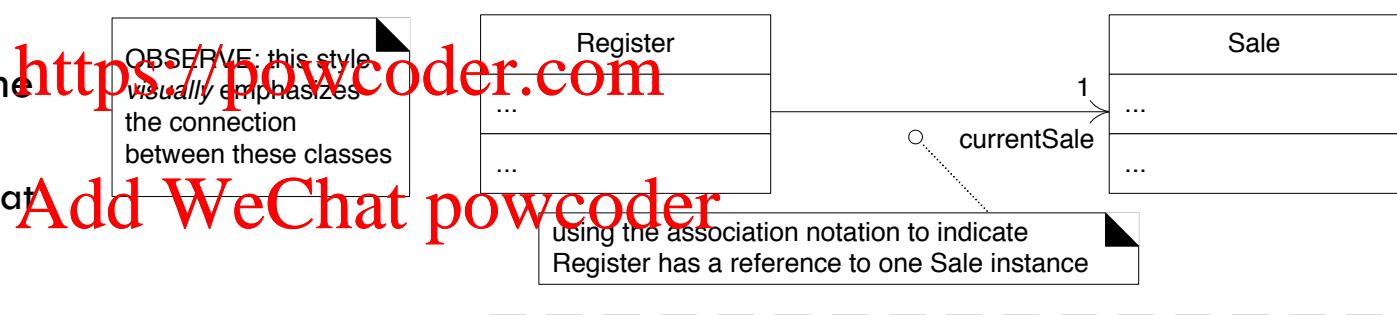


Attribute-as-association

Arrow pointing from the source to the target

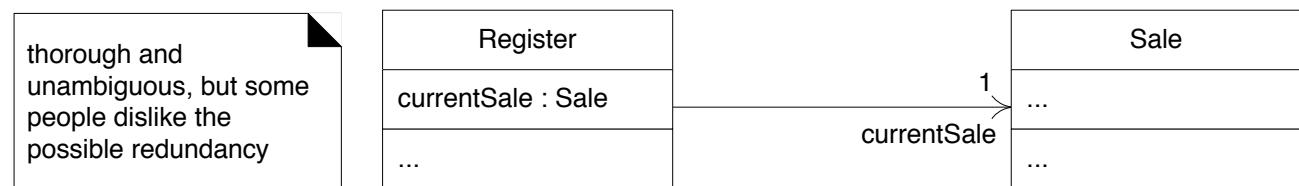
Multiplicity and 'rolename' ('currentSale') at the target

No association name

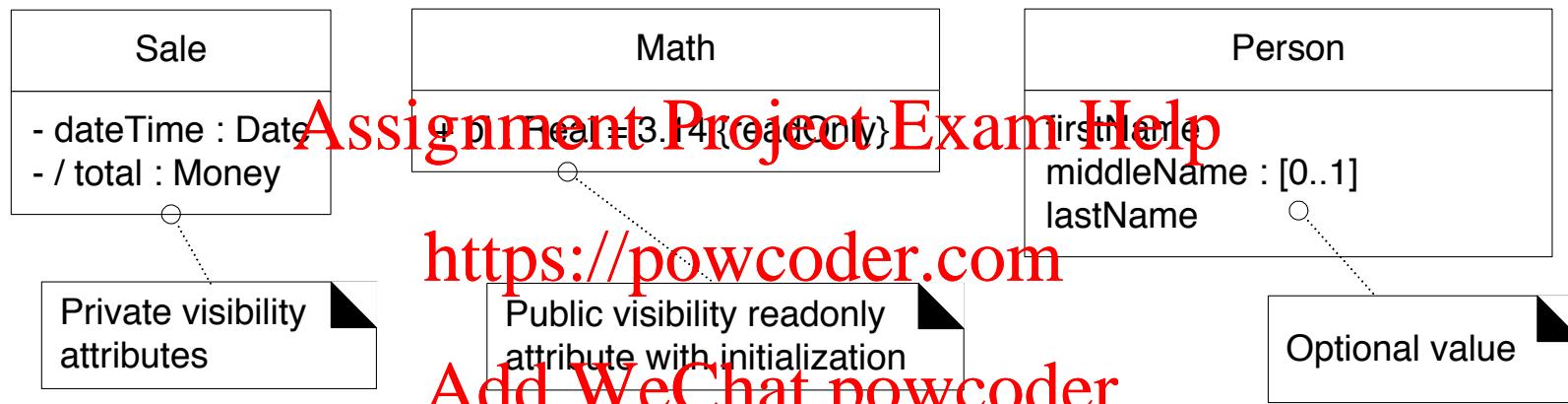


Attribute text and as-association

- Not popular



Class Diagrams – Attribute Examples



Read-only Attributes with initialization, and optional values

Class Diagrams – Operations

Visibility Name (parameter-list) : return-type {property-string} (UML 1)

Visibility Name (parameter-list) {property-string} (UML 2)

- Not a method, but declaration with a name, parameters, return type, exception list, and possibly a set of constraints of pre-and post-conditions

Assignment Project Exam Help

- Operations are public by default, if no visibility shown

- Operation signature in a programming language is allowed, e.g.,

+ getPlayer (name : String) : Player {exception IOException}

Public Player getPlayer(String name) throws IOException

Class Diagrams – Methods

- Implementation of an operation, can be specified in:
 - Class diagrams using UML note symbol with stereotype symbol «method»
 - Mixing static view (class diagram) and dynamic view (method implementation)
 - Good for code generation (forward engineering)
 - Interaction diagrams by <https://powcoder.com> messages



UML Keywords

- Textual adornment to categorize a model element
 - Using «» or { }
 - UML 2 the brackets («») are used for keywords and stereotype

Assignment Project Exam Help

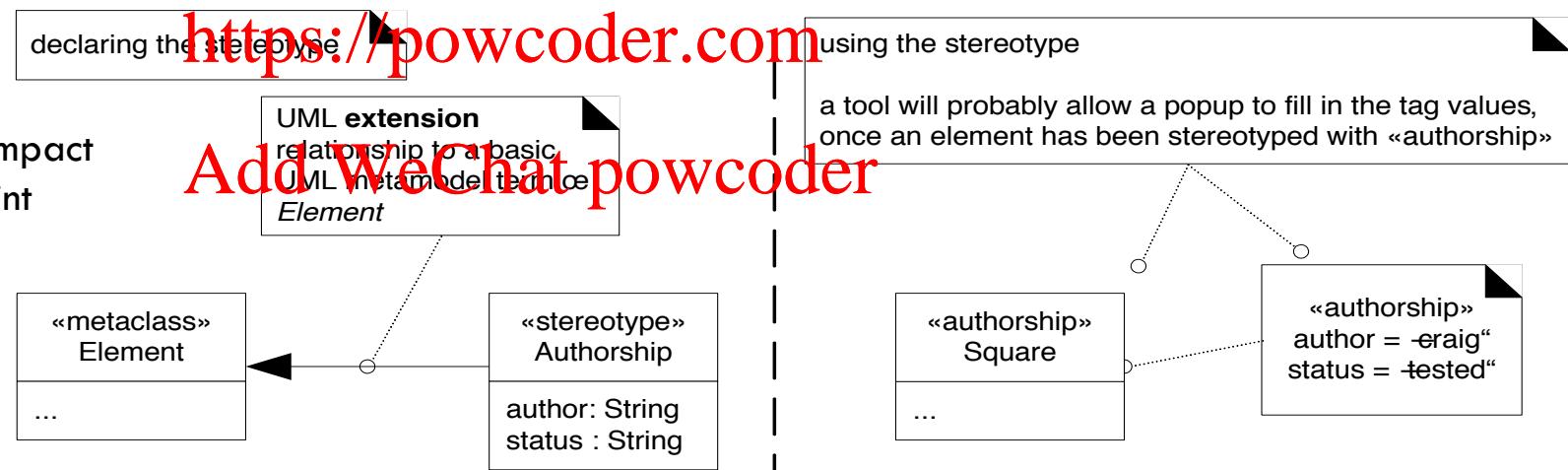
Keyword	Meaning	Example usage
«interface»	Classifier is an interface	In class diagram, above classifier name
{abstract}	Abstract element; can't be instantiated	In class diagrams, after classifier name or operation name
{ordered}	A set of objects have some imposed ordered	In class diagrams, at an association end

UML Stereotypes

- **Stereotypes** allow refinement (extension) of an existing modeling concept
 - Defined in UML Profile
- **UML profile:** group of related model elements allow customizing UML models for a specific domain or platform
 - Extends UML's «metaclass» Element

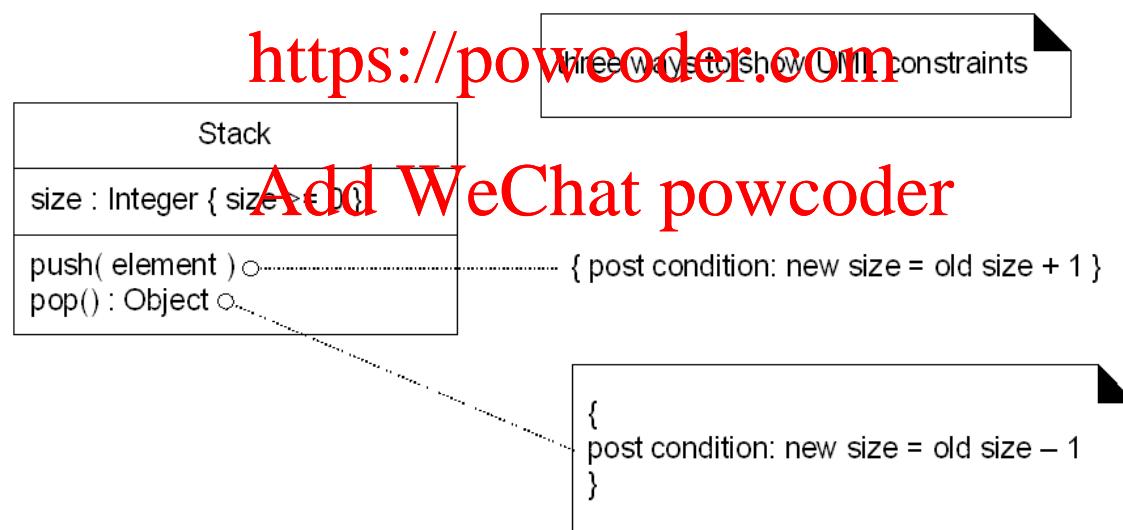
• UML note symbol

- Has no semantic impact
- Specify a constraint



Constraints

- Restriction/condition on a UML elements described in a natural or a formal language (Object Constraint Language (OCL))
- Different ways to represent constraints



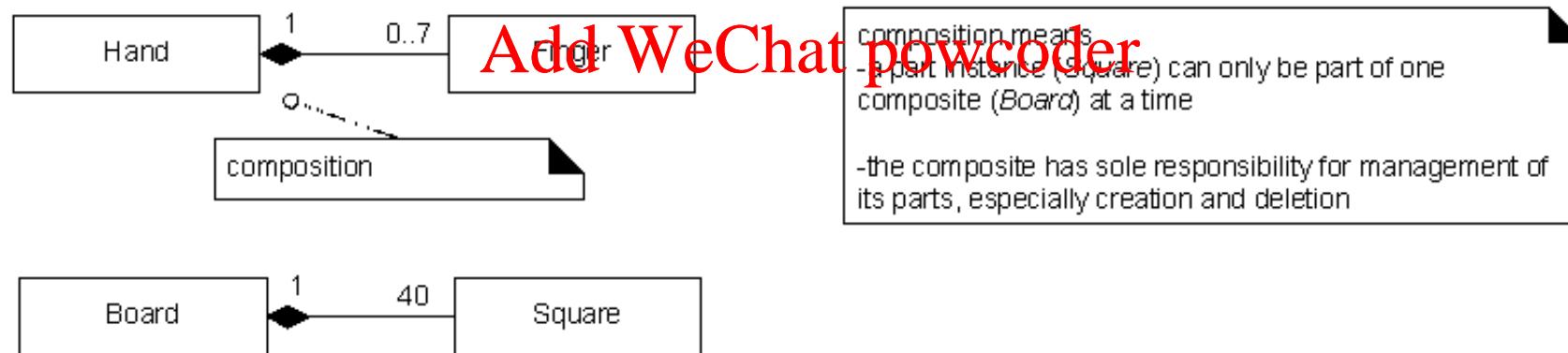
Generalization, Abstract Classes & Operations

“Generalization – a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, specific classifiers indirectly has features of the more general classifiers.” [OMG2003] Assignment Project Exam Help

- Generalization implies inheritance in design class diagram (software perspective) but not in the domain model (conceptual perspective)
Add WeChat powcoder
- Classes and operations with **{abstract}** tag are abstract
- Classes and operations with **{leaf}** are **final** (cannot be overridden in the sub-classes)

Composition

- Composition, or composite aggregation, relationship implies:
 - Instance of the part (e.g., Square) belongs to only one composite instance at a time (e.g., one board)
 - The part must always belong to a composite
 - The composite is responsible for the creation and deletion of its parts (by itself or by collaborating with other objects)



Associations

- Relationship between classifiers where logical or physical link exists among classifier's instances

-"reading direction arrow"
-it has **no** meaning except to indicate direction of reading the association label
-often excluded

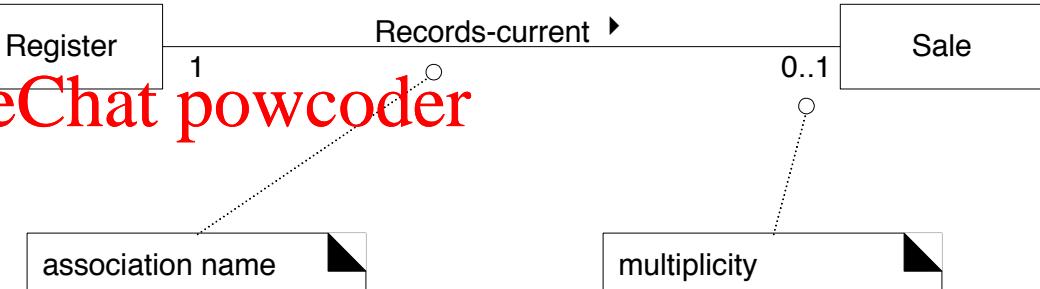
- May implemented differently; no certain construct linked with association

<https://powcoder.com>

- Notations:

- Association name (meaningful)
- Multiplicity
- Direction arrow

Add WeChat powcoder



Associations – Multiplicity

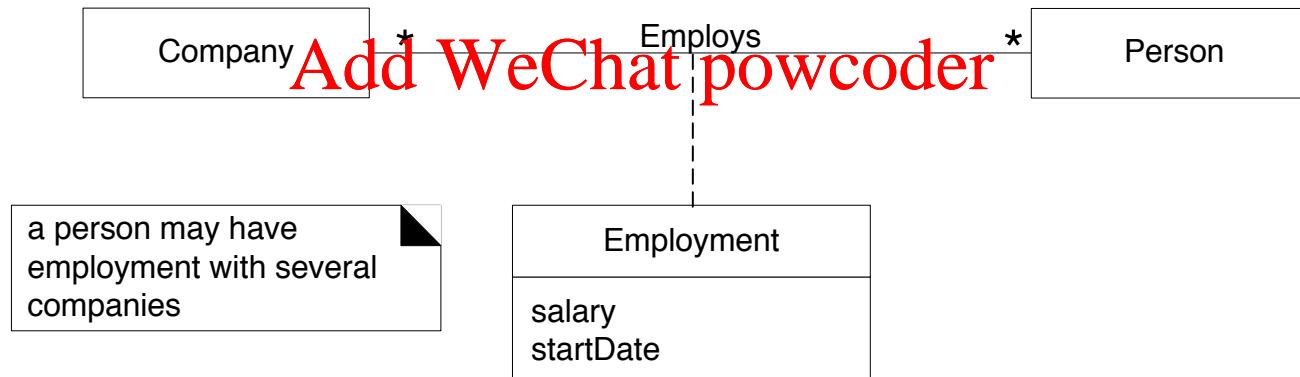
- Multiplicity: number of instances involved in the relationship (association)
- Communicates domain constraints that will be implemented
- Multiplicity focus on the relationship at a particular moment, rather than over a span of time
 - “In countries with monogamy laws, a person can be *Married-to* only one other person at any particular moment, even though over a span of time, that same person may be married to many persons.”

Multiplicity	Meaning (number of participating instances)
*	Zero or more; many
0..1	Zero or one
1..*	One or more
1..n	One to n
n	Exactly n
n, m, k	Exactly n, m or k

Association Class

- Modeling an association as a class (with attributes, operations & other features)
 - A Company *Employs* many Persons
 - *Employs* → Employment class with attributes: salary and startDate

<https://powcoder.com>



Dependency

- A dependency exists between two elements if changes to the definition of one element (the supplier) may cause changes to the other (the client)
- Various reason for dependency
 - Class send message to another <https://powcoder.com>
 - One class has another as its data [Add WeChat powcoder](#)
 - One class mention another as a parameter to an operation
 - One class is a superclass or interface of another

When to show dependency?

- Be selective in describing dependency
- Many dependencies are already shown in other format
- To depict global, parameter variable, local variable and static-method
- To show how changes in one element might alter other elements

- There are many varieties of dependency, use keywords to differentiate them
- Different tools have different sets of supported dependency keywords:

Add WeChat powcoder

<<call>> the source calls an operation in the target

<<use>> the source requires the targets for its implementation

<<parameter>> the target is passed to the source as parameter.

Dependency Example

the *doX* method invokes the *runFinalization* static method, and thus has a dependency on the *System* class



Assignment Project Exam Help

<https://powcoder.com>
Add WeChat powcoder

```
1 public class Foo{  
2     public void doX(){  
3         System.runFinalization();  
4         //...  
5     }  
6 }  
7 }
```

UML Interaction Diagrams

Dynamic (Behavioural) Diagrams

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



UML Interaction Diagrams

- One of the dynamic (behavioral) diagrams which consists of diagrams including Sequence and Communication diagram

Sequence diagrams: illustrate sequence/time-ordering of messages in a fence format (each object is added to the right)

Communication diagrams: objects' interactions are illustrated in a graph/network format

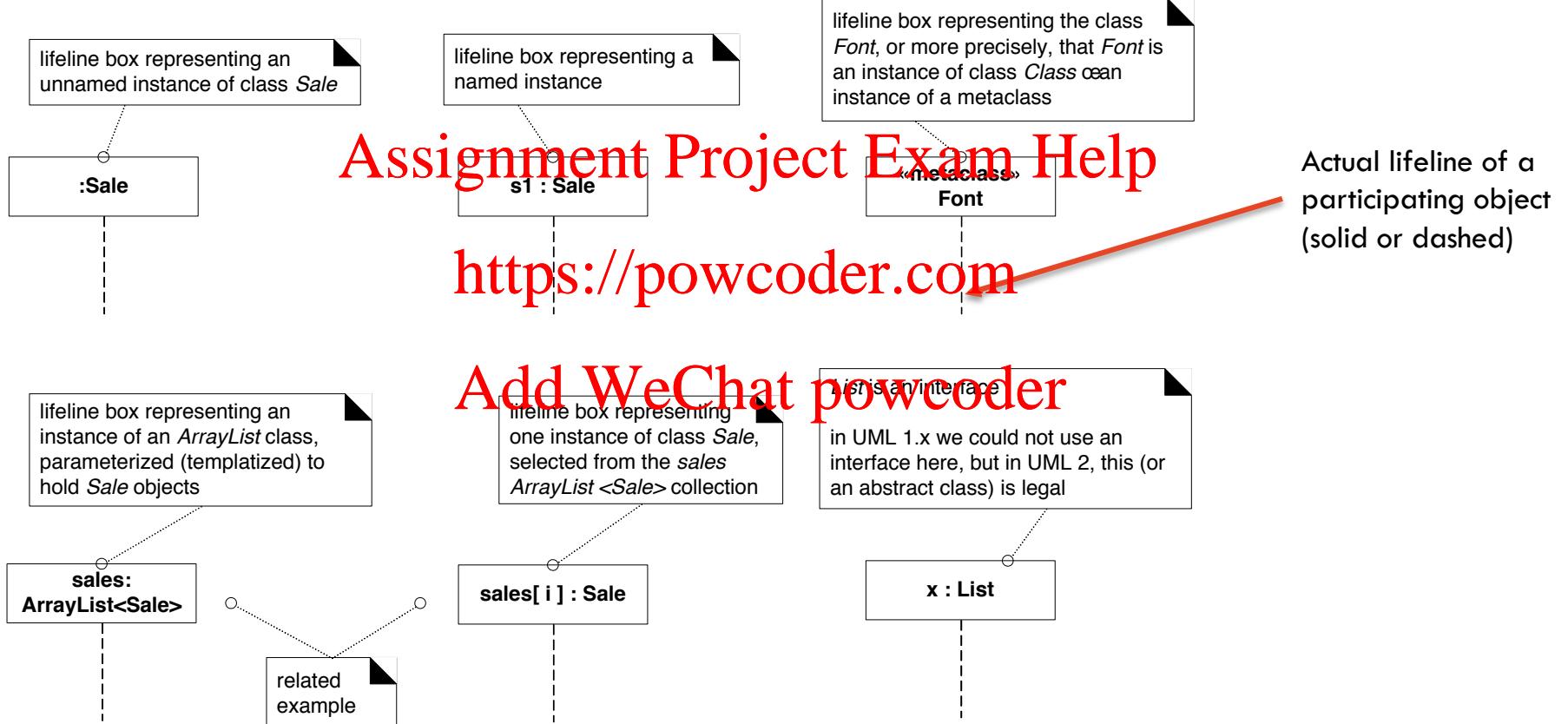
Assignment Project Exam Help

<https://powcoder.com>



Sequence Diagrams: Classes/Objects

- Participants in interactions (class instances, lifeline boxes)



Sequence Diagrams: Messages

- Standard message syntax in UML

ReturnVar = message (parameter : parameterType) : returnType

Assignment Project Exam Help

- Some details may be excluded.

Examples:

Initialize(code)

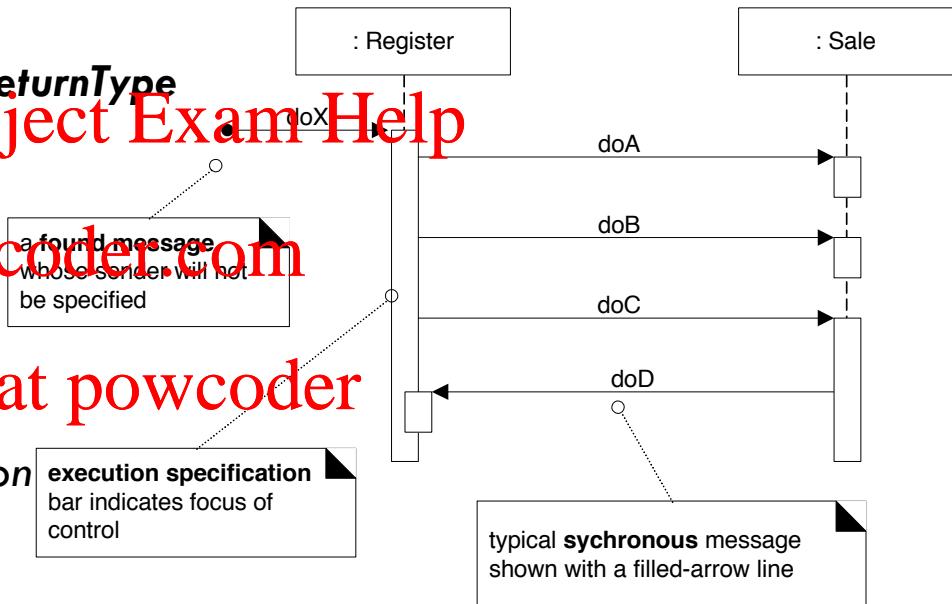
descr = getProductDescription(id)

descr = getProductDescription(id) : ProductDescription

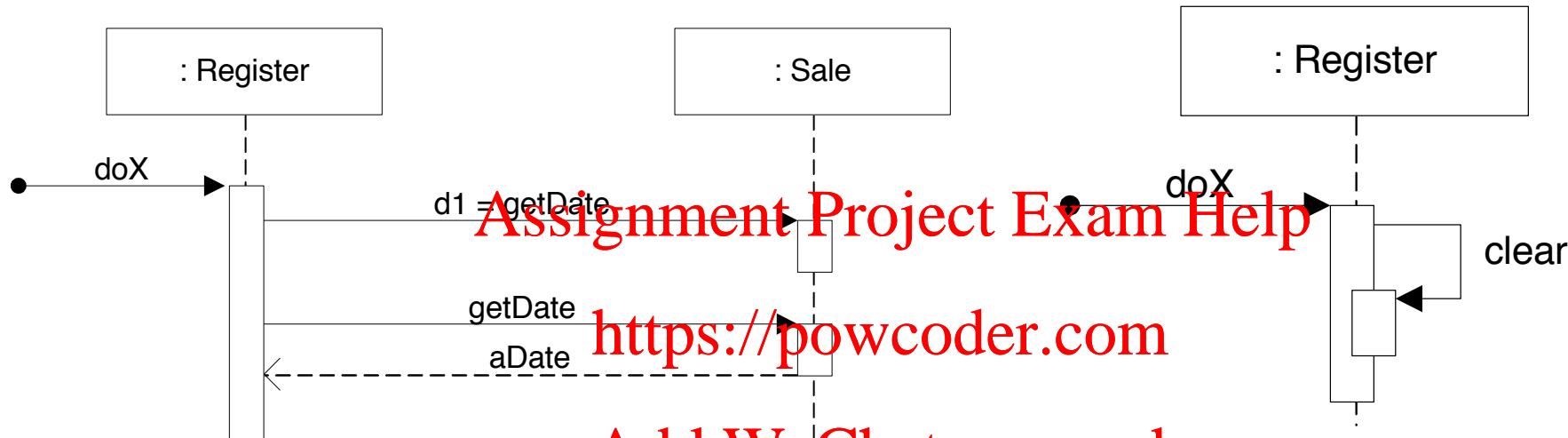
- The time ordering from top to bottom of lifelines

<https://powcoder.com>

Add WeChat powcoder



Sequence Diagrams: Messages



Message reply/return

1. Standard reply/return message syntax

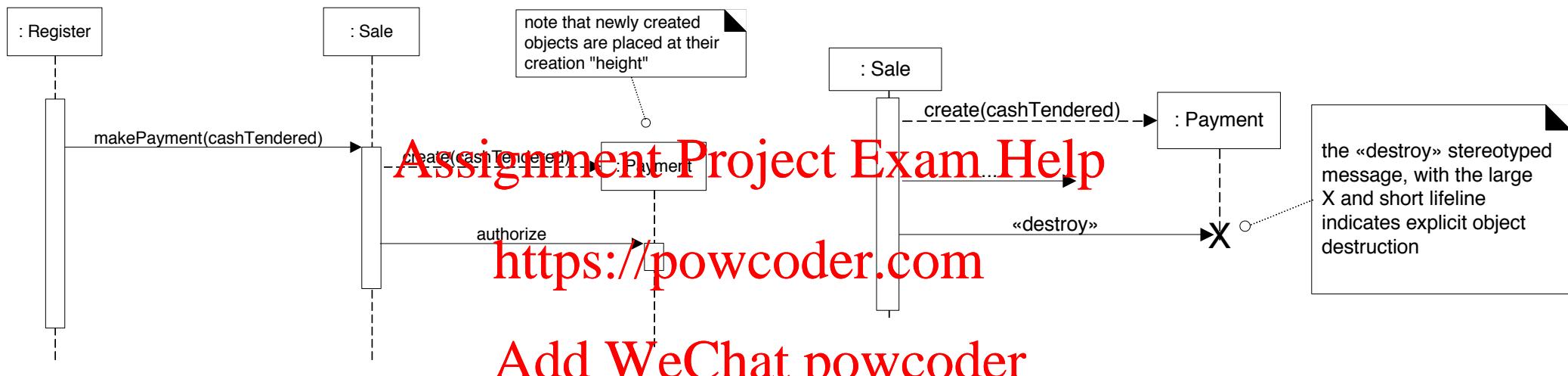
`returnVar = message (parameter)`

2. Reply/return message line at the end of execution bar

Messages to “Self”

Using nested execution bar

Sequence Diagrams: Objects Creation/Destruction



Object Creation

Read as:

"Invoke the new operator and call the constructor"

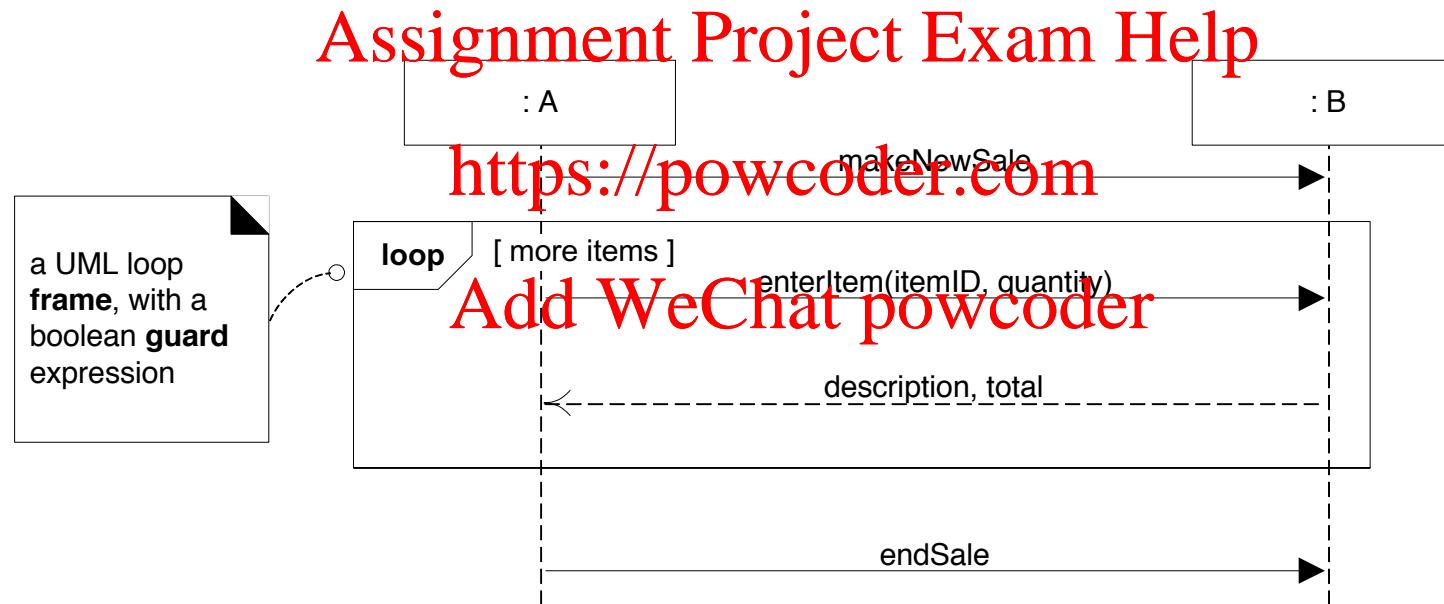
Message name is optional

Object Destruction

Explicit destruction to indicate object is no longer useable (e.g., closed database connection)

Sequence Diagrams: Frames

- Diagram frames in UML sequence diagrams
 - Support conditional and looping construct



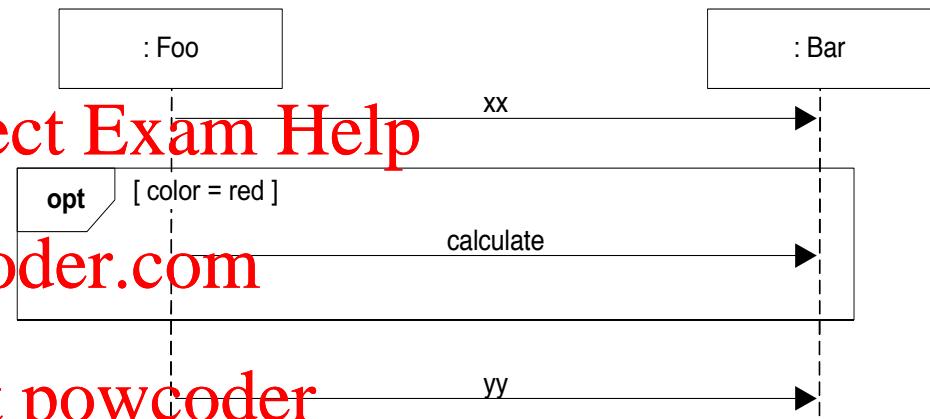
Sequence Diagrams: Frames

- Common frame operators

Frame operator	Meaning
Alt	Alternative fragment for mutual exclusion conditional logic expressed in the guards https://powcoder.com
Loop	Loop fragment while guard is true
Opt	Optional fragment that executes if guard is true
Par	Parallel fragments that execute in parallel
Region	Critical region within which only one thread can run

Sequence Diagrams: Conditional Messages

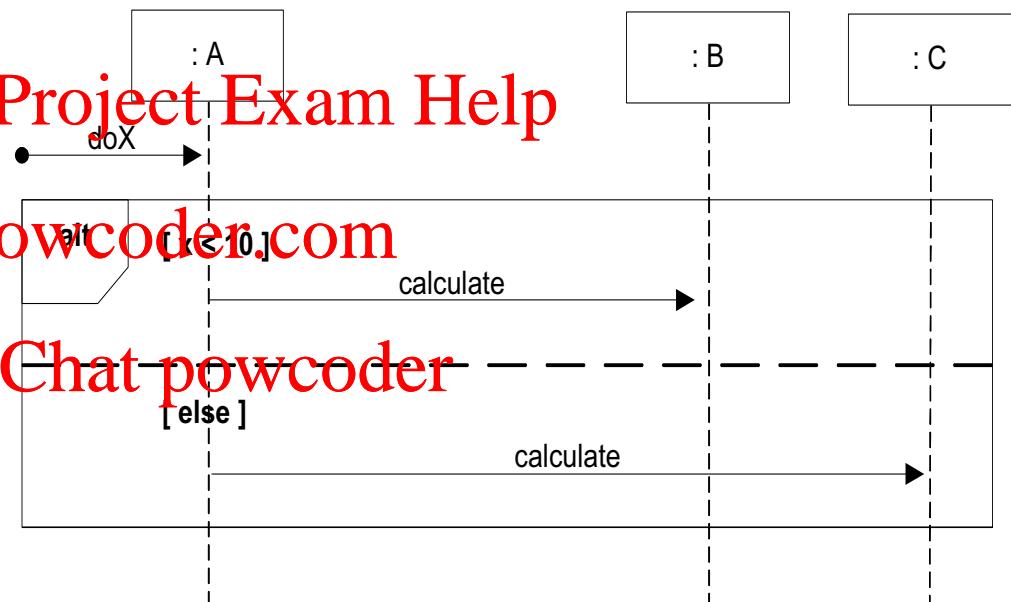
```
public class Foo {  
    Bar bar = new Bar ( );  
    ...  
    public void ml ( ) {  
        bar.xx( );  
        if (color.equals("red"))  
            bar.calculate( );  
        bar.yy( );  
    }  
}
```



Sequence Diagrams: Conditional Messages

Mutually exclusive conditional messages

```
1 public class A{  
2     B b = new B();  
3     C c = new C();  
4     public void doX(){  
5         ...  
6         if (x < 10)  
7             b.calculate()  
8         else  
9             c.calculate();  
10    }  
11 }  
12 }
```

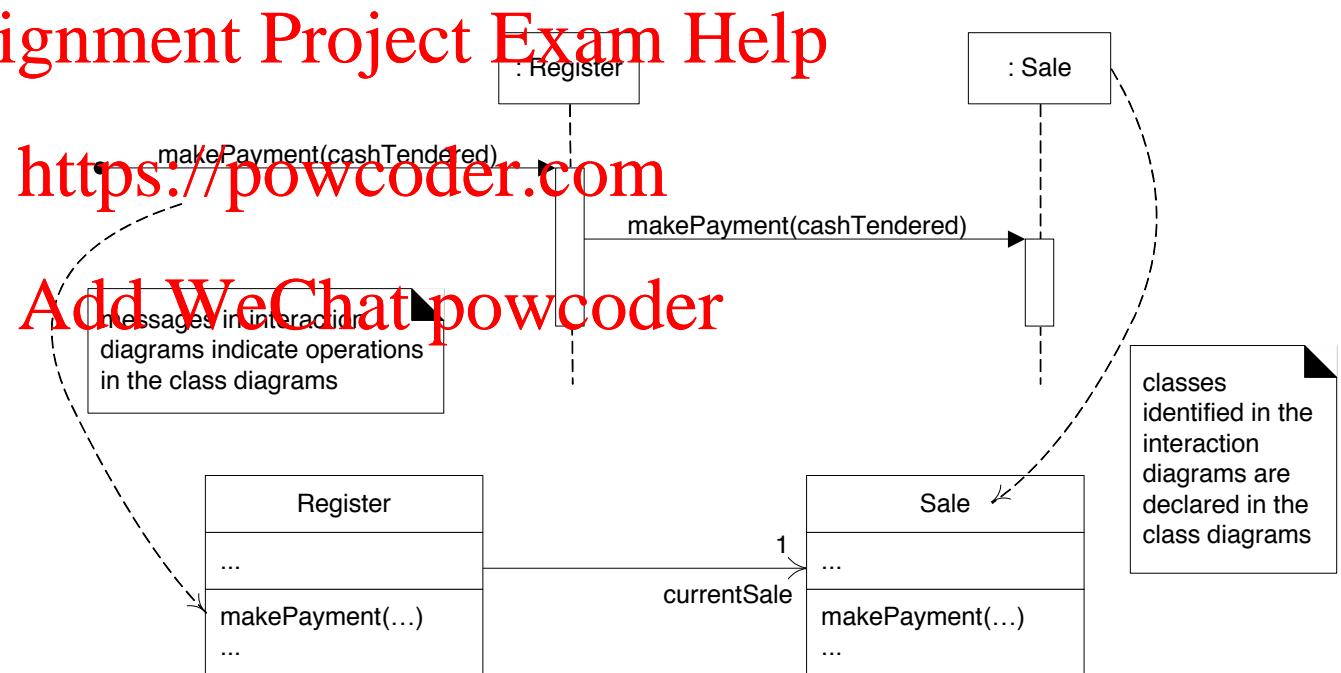


Sequence Diagrams: Nesting of Frames



Class Diagrams: Relationship to Interaction Diagrams

- Interaction diagrams illustrates how objects interact via messages (dynamic behavior)
 - Classes and their methods can be derived
 - E.g., **Register** and **Sale** classes from *makePayment* sequence diagram
- Agile modeling practice:
draw diagrams
concurrently as dynamic
and static views
complement each other
during the design process



Software Modelling Case Study

Assignment Project Exam Help

<https://powcoder.com>

NextGen POS software modeling

Add WeChat powcoder

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



Next Gen Point-of-Sale (POS) System

- A POS is a computerized application used (in part) to record sales and handle payments

Assignment Project Exam Help

- Hardware: computer, bar code scanner
- Software <https://powcoder.com>
- Interfaces to service applications: tax calculator, inventory control
- Must be fault-tolerant (can capture sales and handle cash payments even if remote services are temporarily unavailable)
- Must support multiple client-side terminals and interfaces; web browser terminal, PC with appropriate GUI, touch screen input, and Wireless PDAs
- Used by small businesses in different scenarios such as initiation of new sales, adding new line item, etc.



Add WeChat powcoder

Next Gen POS Analysis

Assignment Project Exam Help

<https://powcoder.com>

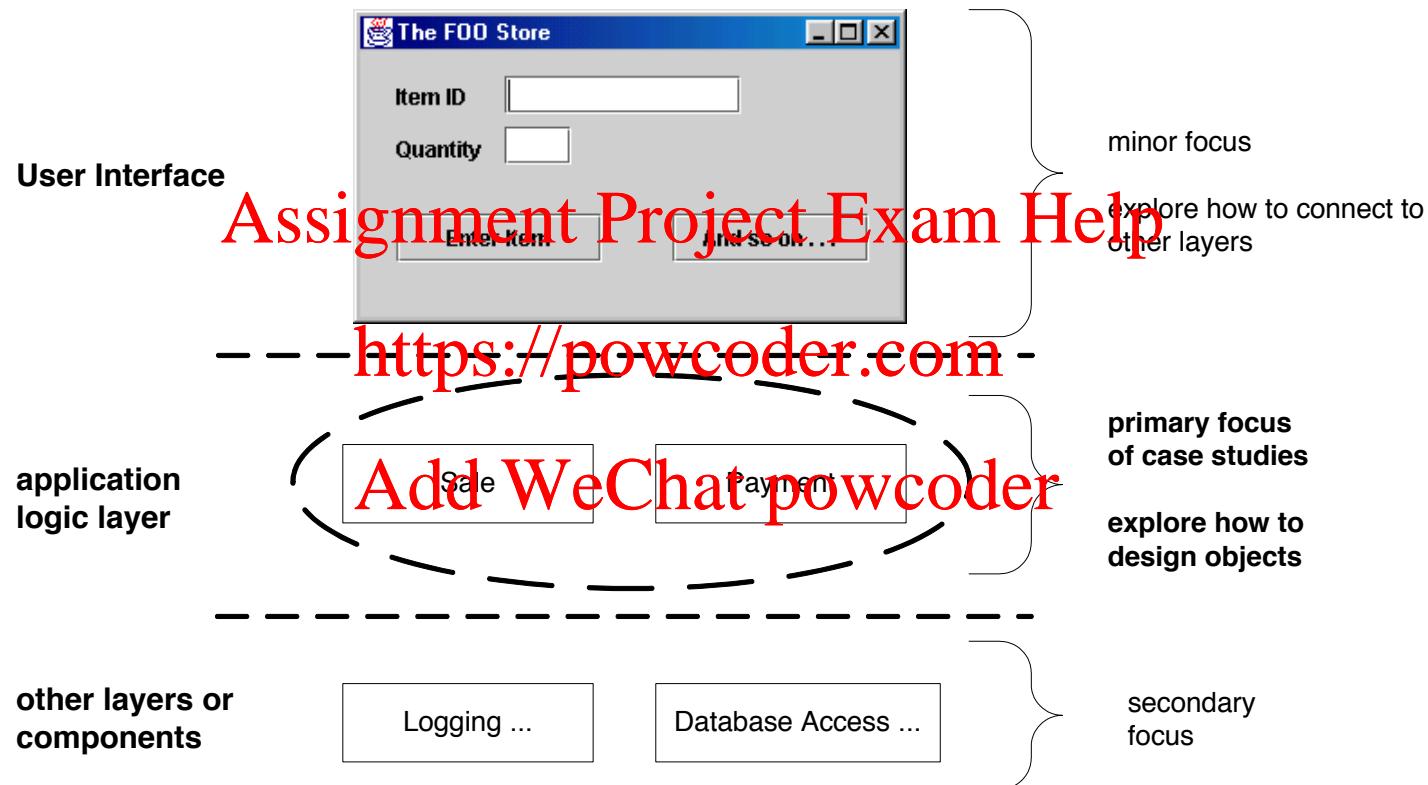
Scope of OOA & D and Process Iteration

Add WeChat powcoder

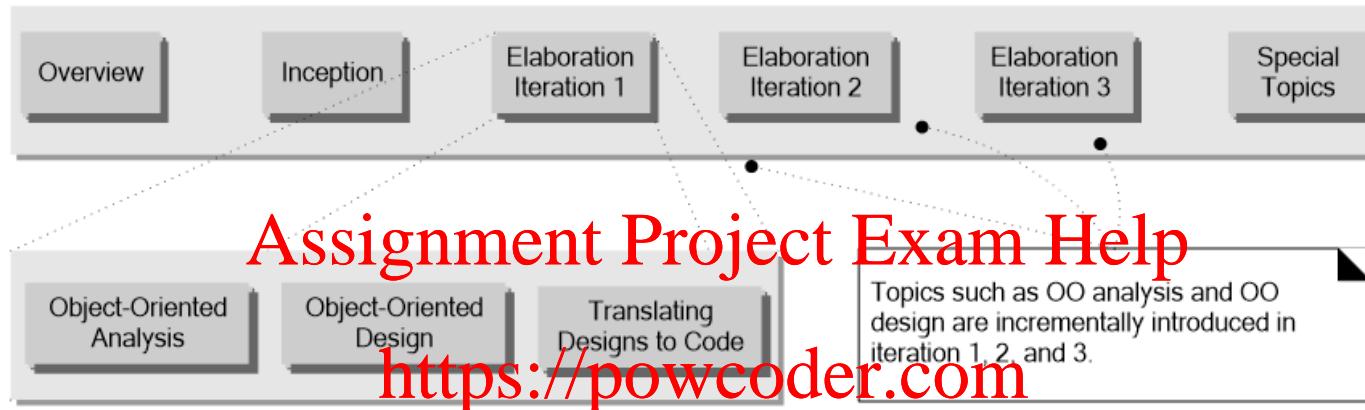
Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



Next Gen POS – Scope (Analysis & Design)



Iteration and Scope – Design and Construction



Add WeChat powcoder

Iteration 1

Introduces just those analysis and design skills related to iteration one.

Iteration 2

Additional analysis and design skills introduced.

Iteration 3

Likewise.

Next Gen POS: From Analysis to Design

Requirements Analysis (OOA)

Business modelling – Domain Models

Use case diagrams

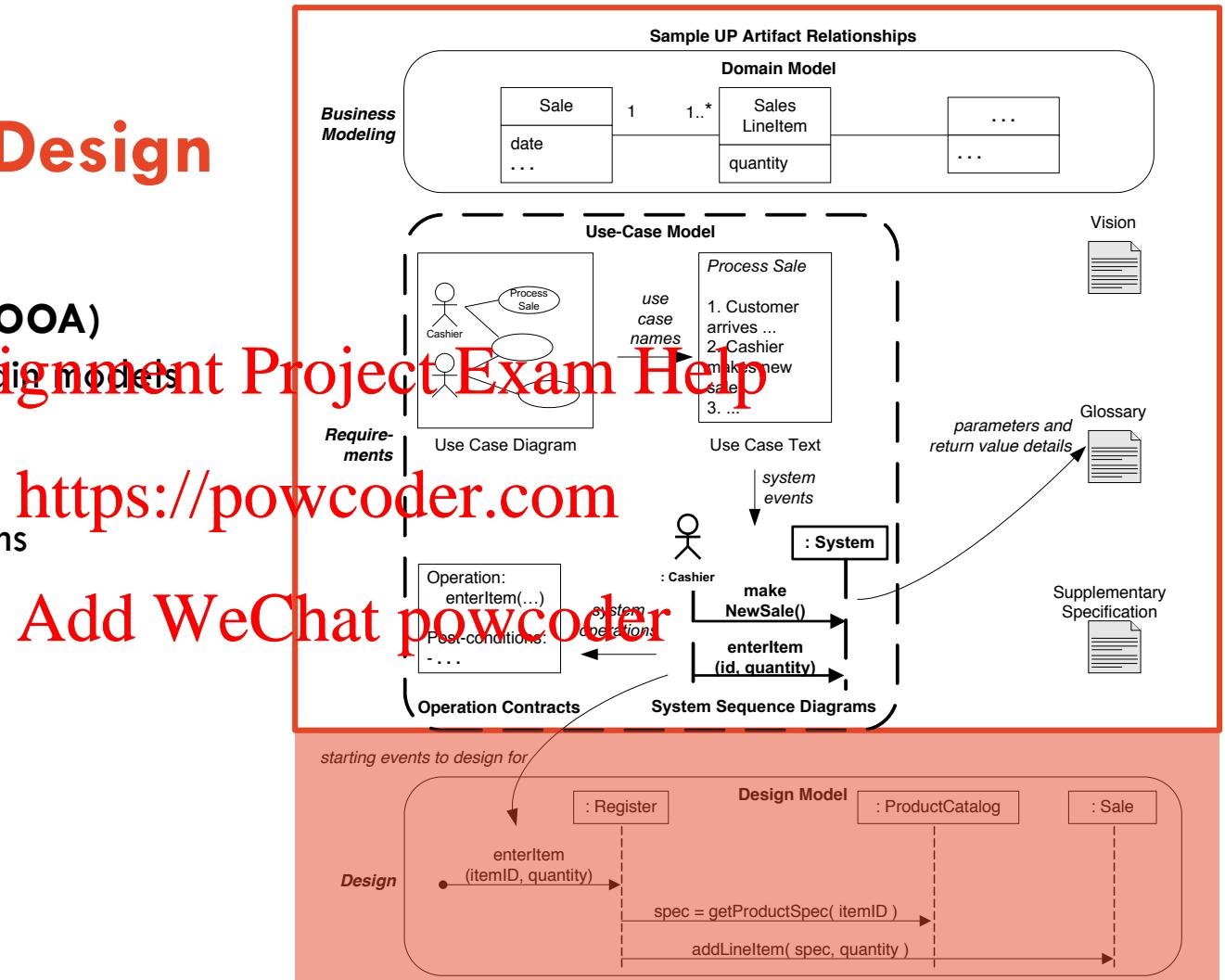
Use case description

System Sequence Diagrams

Design (OOD)

Sequence diagrams

Class diagrams



Self Learning Case Study

1. Use Cases
2. Use Case Diagram
3. Domain Model
4. Class Diagram
5. Interaction Diagram

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition).



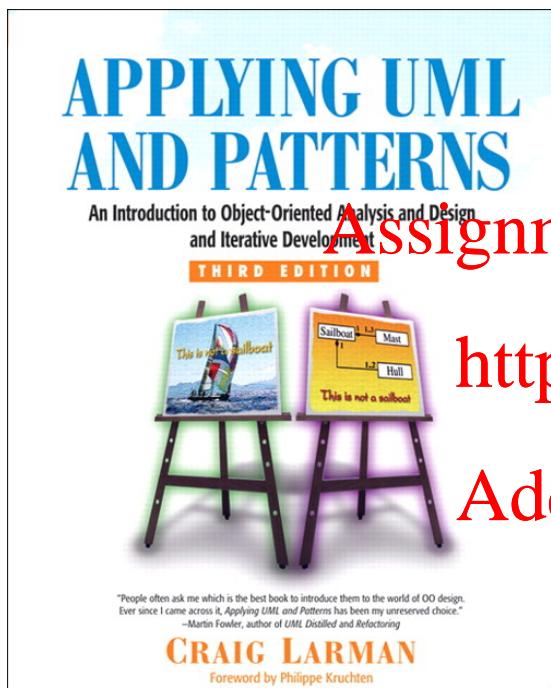
Task for Week 3

- Submit weekly exercise on canvas before 23.59pm Sunday
- Self learning on Next Gen POS system
- If you haven't started your assignment 1, it would be a best time to start it from now on.

<https://powcoder.com>

Add WeChat powcoder

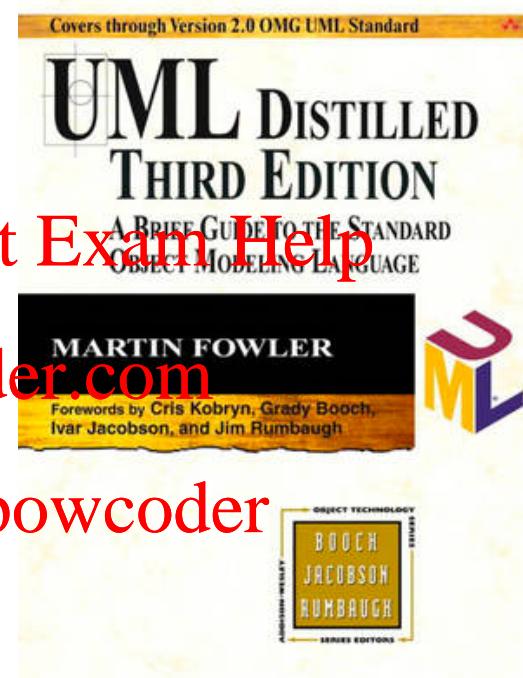
References



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



[Link to USYD Library](#)

[Link to USYD Library](#)

What are we going to learn next week?

- Software Design Principles

- Design Smells
- SOLID
- GRASP

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder