Year 2 Computing/ISDF
Object Oriented Analysis & Design

Team Project – Semester 1 2021-22

Academic Honesty

By submitting your project for assessment you agree to the following:

"The material contained in this assignment is original work by me and my assigned team members, except where work is clearly identified and duly acknowledged. No aspect of this assignment has been previously submitted for assessment in any other unit or course."

For your assigned project case study answer the following questions, submitting:

- Part 1: DIAGRAMS: a 3-page PDF document
  - Page 1: Cover page listing students in team
  - Page 2: Description of teamworking
    - Team member roles / contributions to each deliverable
    - Evidence / description of how you have worked as a
      For example: use of online team management tool (if you used one) / communication methods / record of meetings / online shared documents …
  - Page 3: Use Case Diagram
  - Page 4: Class-Object-Package Diagram
  - **DEADLINE: Week 9**
    Moodle submission 1pm Friday 26th November 2021
    - **Every member of the team must submit the same PDF**

- Part 2: CODE: a ZIP folder
  - PDF: Cover page listing students in team, and their roles/contributions
  - folder "java" Java code
  - folder "php" PHP code
  - NOTE:
    - You may also include an improved 1-page PDF Class-Object-Package Diagram
    - this means you can fix problems in your diagram that come to light when implementing, and so not be forced to implement a poor software design
  - **DEADLINE: Week 11**
    Moodle submission 1pm Friday 10th December 2021
    - **Every member of the team must submit the same ZIP**

Project grading / defence by team members: your timetabled Week 12 lab session

- Every member of the team must be present to answer questions/demonstrate their understanding of their claimed contributions
  - If you are not present you won't get a grade (excepting medical certs…)
- You may be asked to explain / amend diagrams and code components
- 20-30 minutes per team

To demonstrate that this is a team submission:

- every member of the team needs to submit the same documents.
- So ensure you work as a team to complete and share the final versions between yourselves before the upload deadline…

General Project assessment criteria:

- Teamwork
- Quality & relevance/real world plausibility to the assigned Case Study
- Correctness
- Consistency
- Completeness
- Demonstration of full range of module topics
    - (State Machine Diagram topic not required)

NOTE:

- You need to ensure the <u>size and complexity of your OO system design</u> is appropriate for the number of members of your team
    - So the work for a 4-person team would be less than for a 6-person team
    - Ensure you add enough use cases and classes so that everyone on the team has software components they can be responsible for, and defend
    - It should be a coherent software SYSTEM
        - But have components and packages students work in pairs and individually on
- "we all worked on diagram X / class Y together" is not acceptable
    - The team has a set of DELIVERABLES
    - Members of the team need to be responsible for creating versions of the deliverables
    - E.g.
        - Matt will create a first draft of the use case diagram for week 7
        - Joanne will create first draft of Java code for classes A/B/C for week 9
        - Sean will create the final version of the class diagram to present to the team 3 days before the final deadline for final checking …
        - And so on …

Part 1: Class Diagram

Draw a class diagram for the given case study

- there should be somewhere in the class diagram to store the data and relationships to enable and record all system behaviours described in the case study

Show appropriate:

- classes
- associations
    - inheritance
    - association between objects of classes

For <u>every</u> class show:

- attributes and visibility
- several plausible operations

For <u>at least one</u> class show:

- all attributes with private or protected visibility
- public getter and setter operations for that attribute

For <u>every</u> non-generalisation association show:

- name and reading direction (for one direction)
- multiplicities
- named property or array to implement the association

Demonstrate <u>at least one</u> of each the following in your diagram:

- an abstract class
- an enumeration class & and its use by at least one attribute in your diagram
- (but demonstrate more than one if appropriate)

For a top grade you should also demonstrate the following in your diagram:

- interfaces
- overloading of methods

Part 1: Use Case Diagram

Draw a use case diagram for the given case study.

Ensure you illustrate any relevant:

- system name and boundaries
- actors
- use cases
- associations between actors and use cases
- associations between actors and actors
- associations between use cases and use cases

Include at least one of each the following in your diagram:

- generalisation association between actors and actors
- generalisation association between use cases and use cases
- include associations between use cases and use cases
- extend associations between use cases and use cases
    - provide a condition for any extending use case

Part 2: Java code (folder "java")

In Java implement ALL classes and enumeration classes from your class diagram, to demonstrate the following OOP concepts:

- constructors
- accessor methods (getters and setters)
- packages
- enumerations
- inheritance

NOTE:

- You may have empty method bodies for some methods
- But do fully implement all constructors / toStrings / accessor methods

Include a **Main.java** class to:

- create and display properties of at least two instance objects of each of your implemented classes
    - and populate many properties of each object
- demonstrate sending messages to the objects to invoke behaviours
- demonstrate creation associations between objects
    - implementing associations between objects from your diagram

For a good grade you should also demonstrate the following in your implementation:

- exceptions
- interfaces
- overriding of inherited methods using super()

Part 2: PHP code (folder "php")

In PHP implement at least 4 classes from your class diagram, to demonstrate the following OOP concepts:

- constructors
- accessor methods (getters and setters)
- inheritance

NOTE:

- You may have empty method bodies for some methods
- But do fully implement all constructors / toStrings / accessor methods

Include a **main.php** script to:

- creates at least two instance objects of each of your implemented classes
    - and populate many properties of each object
- demonstrate sending messages to the objects to invoke behaviours
- demonstrate creation associations between objects
    - implementing associations between objects from your diagram

For a good grade you should also demonstrate the following in your implementation:

- a simple autoloader script

NOTE:

- code should be error free!!
- Java code should compile, and Main.**main()** should create and output object values
- PHP code should run error free, and **main.php** should create and output object values

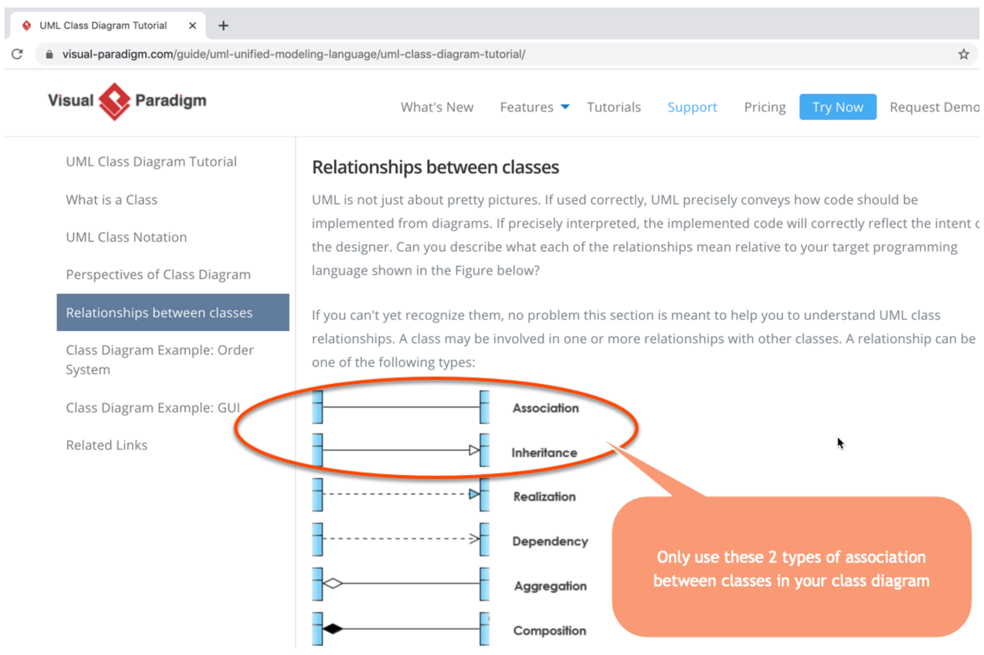
APPENDIX: Hints/Notes for Class Diagrams

- Arrows:
  - the only arrows in your diagram should be for <u>inheritance</u> associations

- classes can appear more than once in the diagram – they are the same class
  - so you can avoid over overlapping lines in your diagram

- relationship connectors – only use these 2 types of class relationship in your diagram:
  - class-to-class association
    - solid line (no arrows)
  - one class is generalisation of another class
    - white triangle solid arrow

- do <u>not</u> draw any of the following connectors in your diagram:
  - ***realization / dependency / aggregation / composition***


UML Class Diagram Tutorial
visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
Visual Paradigm
What's New
Features
Tutorials
Support
Pricing
Try Now
Request Demo
UML Class Diagram Tutorial
What is a Class
UML Class Notation
Perspectives of Class Diagram
Relationships between classes
Class Diagram Example: Order System
Class Diagram Example: GUI
Related Links

Relationships between classes

UML is not just about pretty pictures. If used correctly, UML precisely conveys how code should be implemented from diagrams. If precisely interpreted, the implemented code will correctly reflect the intent c the designer. Can you describe what each of the relationships mean relative to your target programming language shown in the Figure below?

If you can't yet recognize them, no problem this section is meant to help you to understand UML class relationships. A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

Association
Inheritance
Realization
Dependency
Aggregation
Composition
Only use these 2 types of association between classes in your class diagram

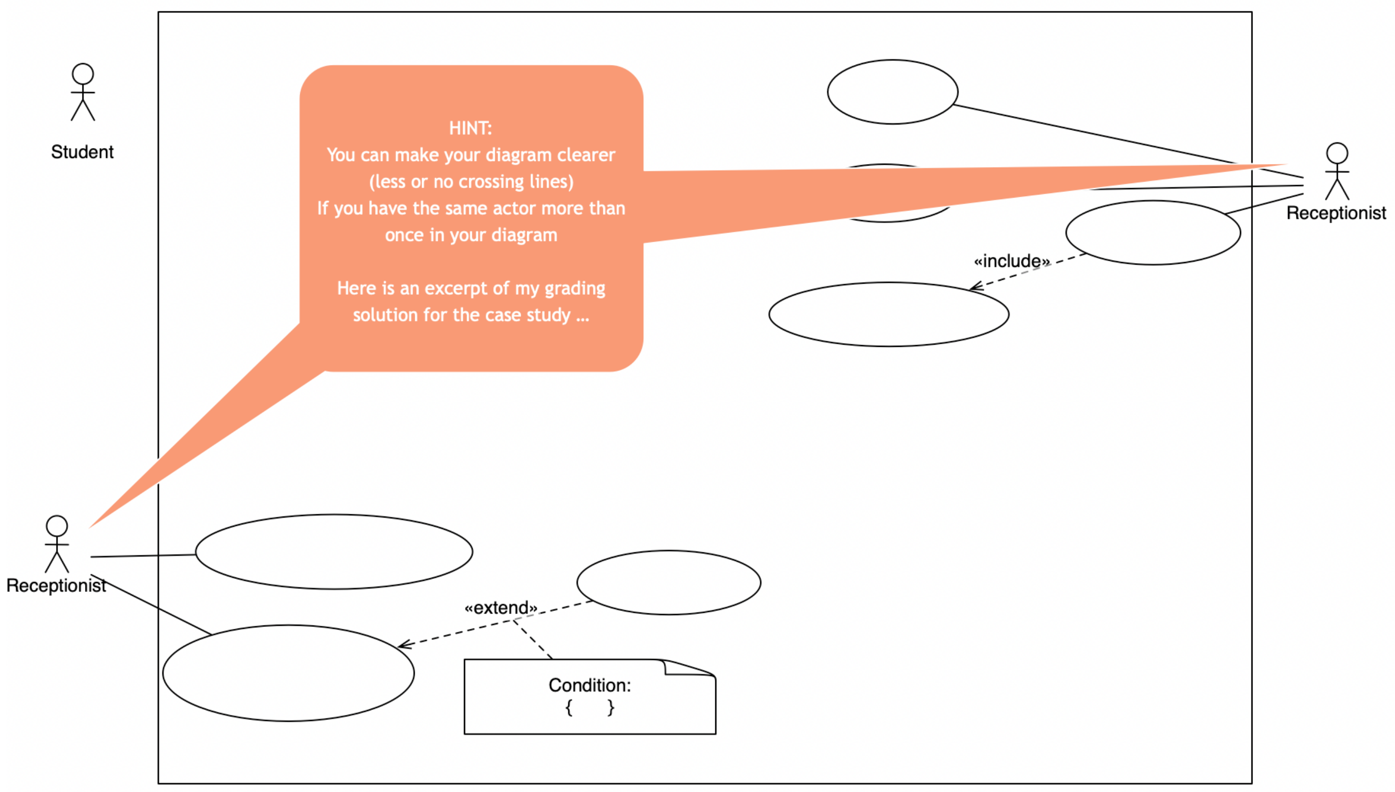
- (Advanted diagrams going for top grade only):
  - If you are showing class implementation of interfaces
  - Please the “lollipop” notation (not the dashed arrow)

APPENDIX: Hints/Notes for Use Case Diagrams

- arrows should only appear in your diagram for the following associations:
  - inheritance between actors
  - inheritance between use cases
  - extend and include associations between use cases
- and ensure you use the correct type of arrow head in each case

- there should not be any arrow on associations between actors and use cases

- actors can appear more than once in the diagram – they are the same actor
  - so you can avoid over overlapping lines in your diagram


Student
HINT:
You can make your diagram clearer
(less or no crossing lines)
If you have the same actor more than
once in your diagram
Here is an excerpt of my grading
solution for the case study ...
Receptionist
«include»
Receptionist
«extend»
Condition:
{ }