Year 2 Computing/ISDF
Object Oriented Analysis & Design

Individual or Team Project – 2023

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 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 agreed project case study answer the following questions, submitting:

- **Deliverable 1: Case Study Actors-Use Cases text summary**
  - Submit: PDF max 2 pages "**actor-use-cases.pdf**"
- **Deliverable 2: Class-Package Diagram (without exceptions)**
  - Submit: PDF 1-page "**class-package-diagram.pdf**"
- **Deliverable 3: Java code Implementation (without exceptions)**
  - Submit: 1 ZIP folder "**java.zip**"

- **(teams only): Deliverable 4 Team meetings & contributions to project**
  - Submit: PDF "**team_meetings_contributions.pdf**"
  - 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…

- **('A' grade attempt only):**
  - **Deliverable 5: Class-Package Diagram (with exceptions)**
    - Submit: PDF 1-page **"class-package-diagram2_exceptions.pdf"**
  - **Deliverable 6: Java code Implementation (with exceptions & db & foreign keys)**
    - Submit: 1 ZIP folder "**java2_exceptions_db.zip"**

DEADLINE for all deliverables:
Week 11: Moodle submission: 2pm Friday 1st December 2023

Project grading & defence by team members: Week 12/13 lab/lecture sessions

- You'll be asked to explain / amend diagrams and code components
  - Around 10 minutes per project
- Sign up on Moodle for a grading session
  - Only 1 member of a team needs to sign up

You are to create a novel project to demonstrate all the topics from the module
Including:

- Classes & inheritance & object associations
- Constructors and constructor chaining
- properties & visibility modifiers
- accessor methods (getters and setters)
- packages
- enumerations
- static vs instance members
- class hierarchies (including abstract and final classes and methods)
- Interfaces, Exceptions, Java database programming

General Project assessment criteria:

- Quality & relevance/real world plausibility to the assigned Case Study
- Correctness
- Consistency
- Completeness
- Demonstration of full range of module topics

Extra notes for teams:

- You need to ensure the size and complexity of your OO system design is appropriate for the number of members of your 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
- **(teams only): Deliverable 4 Team meetings & contributions to project**
    - Submit: PDF "**team_meetings_contributions.pdf**"
    - This should summarise:
        - When the team met & decisions made each meeting
        - WHO did WHAT and WHEN it was delivered to the team
    - "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 identifiable contents of the deliverables
        - E.g.
            - Fred created a first draft of the use case diagram for week 7
            - Joanne created first draft of Java code for classes A/B/C for week 9
            - Sean created the final version of the class diagram to present to the team 3 days before the final deadline for final checking …
            - And so on …

Deliverable 2: Class-Package Diagram (PDF)

Draw a class-package diagram for your given case study

- You only need to design the Entity classes for the case study
  - i.e. the classes needed to store the data and relationships to enable and record all system behaviours described in the case study
  - so all the data required for each use case needs to be somewhere in one of your entity classes

Show appropriate:

- classes
- associations
  - inheritance
  - association between objects of classes
  - interfaces & exceptions (for higher grade projects)

For every class show:

- attributes and visibility
  - appropriate use of public, private and protected visibility
  - accessor methods

For ever 1-to-many / 1-to-1 association between classes show[1]:

- multiplicities
- named property or array to implement the association
  - don't "hard code" this into the property list – annotate the association line with the property to be implemented to link the objects

Demonstrate the following in your diagram:

- packages
- abstract and final classes
- enumeration classes & their its use as class property data types

For a higher grades you should also demonstrate the more advanced topics from the module, including:

- abstract and final methods
- interfaces
- overloading of methods
- Exceptions

Do NOT list any methods for classes

- so 2 rows only for each c lass

[1] Avoid many-to-many associations – resolve these to 1-to-many

Deliverable 3: Java code (ZIP)

In Java implement ALL classes and enumeration classes from your class diagram.

Include a **Main.java** class (Java) to:

- create and display properties of at least two instance objects of each of your implemented classes
  - and populate all properties of each object
- demonstrate sending messages to the objects to invoke behaviours
- demonstrate creation of associations between objects
  - implementing associations between objects from your diagram
- use **toString()** methods to print out useful information about the properties of each object you have created

For a higher grade you should also demonstrate more advanced concepts such as the following in your implementation:

- interfaces
- exceptions
- reading and writing objects to and from a MySQL database

NOTE:

- code should be error free!!
- you should follow normal conventions for naming of files / classes / class members etc.
- Java code should compile, and run from **Main**.**main()**

APPENDIX: Hints/Notes for Class-Package Diagrams

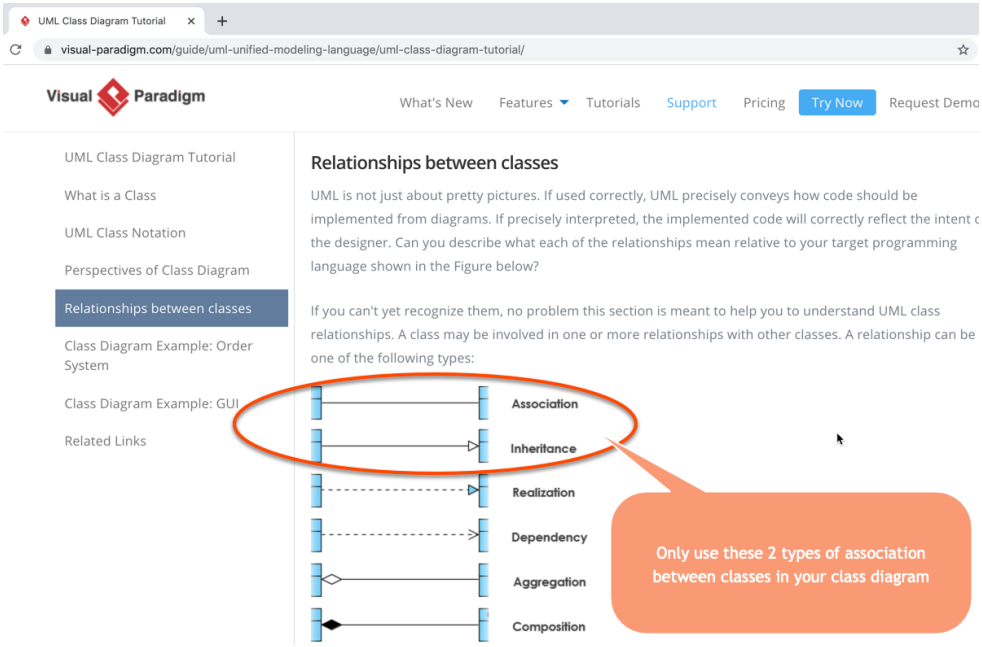
- Arrows:
  - the only arrows in your diagram should be for inheritance associations
  - (and exceptions if gong for 'A' grade)

- 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:
  - generalisation (subclass-superclass)
    - white triangle solid arrow
  - 1-to-1 and 1-to-many association between objects of different classes
    - solid line (no arrows)

  - in addition use the "lollipop" notation for any **interfaces** in your design
    - (do not use the dashed-line open-arrow for **interfaces**)

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



- if showing Interfaces
  - please the "lollipop" notation (not the dashed arrow)

NOTE: Don't over complicated your diagram
- Do NOT list any methods for classes – so 2 rows only
- Do NOT list any "id" property for classes
  - (for DB implementation we will assume every class has an integer id primary key)
- A class may appear MORE THAN ONCE in a diagram
  - To avoid spaghetti association lines
- I recommend STRAIGHT LINES for associations
  - Not curved
  - Not right angled (although that can work for inheritance sometimes)
- Don't let your diagram tool JOIN association lines – you'll lose marks …..