B.Sc. in Computing
Object Oriented Analysis and Design
Mini-Project – 40%

Individual or Team Project(2 members per team) – 2024

Submission Date: Friday 6th December

Submission:

- **Upload to Brightspace a zipped file containing:**

- **Deliverable 1: PDF "class_package-diagram1.pdf"**
  - class-Package Diagram
- **Deliverable 2: ZIP folder "java.zip"**
  - Java code Implementation
- **Deliverable 3: PDF "project_summary.pdf"**
  - Summary of project and where topics are implemented in the project.
  - Team meetings & contributions to project, ie who did what on the project.

DEMO

You must demo your project in your timetabled lab session week beginning 9th December.
Failure to demo will attract zero marks – **no show no grade**.

This can be an individual or team project but all requirements must be completed regardless of individual or team submission.

Learning Objectives

The purpose of this assignment is for the learner to create a professional OO project.

It is an opportunity for you to showcase all (or most) of the practical skills that you have learned during the semester and integrate them in a cohesive manner.

Use of GEN AI is not permitted for this project.

Late Submission

- A penalty of 20% will be applied to the project for each day it is late up to and including 3 days.
- No project will be accepted after Monday December 9th, unless there are mitigating circumstances.

Assignment Brief

You are to create a 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
- Class hierarchies (including abstract and final classes and methods)
- Interfaces, Exceptions

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 include enough use cases and classes so that all team members have software components they can be responsible for, and defend
  - It should be a coherent software SYSTEM
- **(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.

- Ann created a first draft of the use case diagram for week 7
- Tom created first draft of Java code for classes A/B/C for week 9

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
  - 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 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

Demonstrate the following in your diagram:

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

Do NOT list any methods for classes

- 2 rows only for each class

[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
  - populate all properties of each object
- demonstrate sending messages to the objects to invoke behaviours
- demonstrate creation of associations between objects
  - implement associations between objects from your diagram
- use **toString()** methods to print out useful information about the properties of each object you have created

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

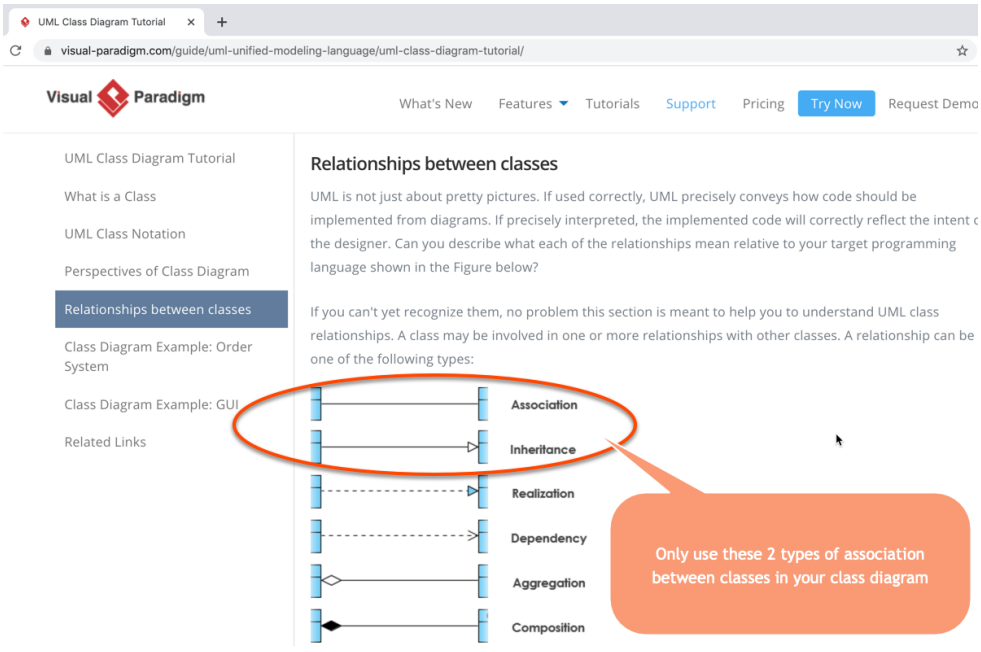
- Arrows:
  - the only arrows in your diagram should be for inheritance associations and exceptions

- 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 the following 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 use the "lollipop" notation (not the dashed arrow)

NOTE: Don't over complicate your diagram

- Do NOT list any methods for classes – so 2 rows only
- Do NOT list any "id" property for classes
- 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

Marking Scheme

A rubric will be used for marking in Brightspace.

Academic Honesty

Any work that you submit for continuous assessment or assignments must contain a significant contribution by you. Any help you receive from someone must be acknowledged in the work submitted. Failure to acknowledge the source of a significant idea or approach is considered plagiarism and not allowed.

Use of GEN AI is not permitted for this project.

Academic dishonesty will be dealt according to the university's plagiarism policy.