Department of Computer Science & Applied Physics

B.Sc. Software Development – Object-Oriented Design Principles & Patterns (2022)
ASSIGNMENT DESCRIPTION & SCHEDULE

Refactoring and Redesigning a Video Game

This assignment constitutes 50% of the total marks for this module.

1. Overview

Isometric projection is a technique for rendering 3D objects in 2D space by equally foreshortening the X, Y and Z axes and maintaining an angle of 120° between them. While isometric projection can be dated back to the 18th century, it became very common in the 19th century as a technical drawing format. In video gaming, the technique allows graphics and sprites developed using 2D Cartesian space to represent a 3D gaming environment, decreasing the space complexity by a full dimension. The first isometric video games emerged in the early 1980's with arcade games such as *Zaxxon* (Sega, 1982) and *Q*bert* (Gottlieb, 1982) and games designed for home computers, such as the ZX Spectrum and Commodore 64. The latter included the groundbreaking isometric games ***Ant Attack*** (Quicksilva, 1983, see http://torinak.com/qaop#!antattack) and the action-adventure game ***Knight Lore*** (Ultimate Play the Game, 1984, see http://torinak.com/qaop#!knightlore).

You are required to refactor, redesign and complete an isometric game that challenges a player to complete some type of quest. The game should end, with suitable celebration and fanfare, when the appropriate player action fulfills the quest. A set of stub classes that implements a fully-functional isometric view and sprites is available on Moodle, but is deliberately badly designed. Your objective is to ***extend***, ***modify*** and ***refactor*** the code provided to create an elegant game design. ***The focus of this assignment is on design, not on implementing more and more features.*** Do not waste your time implementing features and functionality that are

not needed. You are free to redesign and implement functionality in any way that you choose, but you must document and provide a clear rationale for your design. In addition, you should aim to accomplish the following in your application:

- Apply the **SOLID** principles throughout your application.
- Group together **cohesive** elements into methods, types and packages.
- **Loosely-couple** together all programme elements to the maximum extent possible.
- Create **a reusable set of abstractions** that are defined by interfaces and base classes.
- **Encapsulate** at method, type and package level.
- Apply creational, structural and behavioural **design patterns** throughout the application where appropriate.
- Use the **Swing MVC** framework to write custom sprites and threaded game objects.

Note that there is no single "correct" answer to this assignment – there are many possible solutions, all with their advantages and drawbacks. Any design patterns that may apply should already exist in the problem. State any **assumptions or known issues** relating to your design in comments at the top of your classes or in the README file.

You are free to asset-strip any online resources for images and functionality provided that you modify any code used and cite the source both in the README and inline as a code comment above the relevant section of the programme. You are not free to re-use whole components and will only be given marks for work that you have undertaken yourself. Please pay particular attention to how your application must be packaged and submitted and the scoring rubric provided. Marks will only be awarded for features described in the scoring rubric.

2. Deployment and Delivery

The project must be submitted by midnight on 6th January 2023. Before submitting the assignment, you should review and test it from a command prompt on ***a different computer*** to the one that you used to program the assignment.

Do NOT submit the assignment as an Eclipse project or submit any text files or other resources. If you do, you will lose marks. You will also lose marks if you hard-code any environmental variables in your project like system paths and file names.

- The project must be submitted as a Zip archive ***(not a 7z, rar or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the "*Refactoring and Redesigning a Video Game (50%) Assignment Upload*" heading of Moodle. Only the contents of the submitted Zip will be considered. ***Do not add comments to the Moodle assignment upload form.***
- The name of the Zip archive should be <*id*>.zip where <*id*> is your ATU student number.
- You must use the **module name atu.software** and the **package name ie.atu.sw** for the assignment. Make sure that the module is declared as ***open***.

The Zip archive must have the following structure:

Marks	Category
isogame.jar	A Java **archive** containing your API and runner class with a main() method. You can create the JAR file using Ant or with the following command from inside the "bin" folder of the Eclipse project: **jar –cf isogame.jar *** The application should be executable from a command line as follows: **java --module-path ./isogame.jar --module atu.software/ie.atu.sw.Runner**

	You will need to ensure that the module descriptor contains the declaration **exports ie.atu.sw** to execute the command above. You will also need to add the declaration **requires java.desktop;** to the module descriptor to access the Swing API.
resources	A directory structure containing all the resources (images, text etc.) for your application.
src	A directory that contains the packaged **source code** for your application.
README.pdf	A **PDF** file detailing the main features of your application in **no more than 300 words**. All **features and their design rationale** must be fully documented. You should consider also including the UML class diagram and screenshots in this document to help explain your design clearly.
design.png	A UML **class diagram** of your API design. The UML diagram should only show the relationships between the key classes in your design. Do not show private methods or attributes in your class diagram. You can create high quality UML diagrams online at **www.draw.io**.
docs	A directory containing the **JavaDocs** for your application. You can generate JavaDocs using Ant or with the following command from inside the "src" folder of the Eclipse project: **javadoc -d [*path to javadoc destination directory*] ie.atu.sw** Make sure that you read the JavaDoc tutorial provided on Moodle and comment your source code correctly using the JavaDoc standard.

4. Marking Scheme

Marks for the project will be applied using the following criteria:

Element	Marks	Description
Structure	5	**All-or-nothing.** The packaging and deployment are both correct. The application launches correctly without any manual intervention from a command line as specified above.
README	20	All **features and their design rationale** are fully documented in 300 words or less. The README should clearly document where and why any design patterns have been used.
UML	10	A UML class diagram that correctly shows all the **important structures and relationships** between types in your design.
JavaDocs	10	All classes are fully commented using the **JavaDoc** standard and generated docs are available in the *docs* directory.
Robustness	15	The application fully implements the Basic Requirements 1-3.
Cohesion	15	There is very high **cohesion** between modules, packages, types and methods.
Coupling	15	The API design promotes **loose coupling** at every level.
Extras	10	Only **relevant extras** that have been fully documented in the README.

You should treat this assignment as a project specification. Any deviation from the requirements will result in some or all marks not being earned for a category. Each of the categories above will be scored using the following criteria:

Range	Expectation
0–39%	***Not delivering*** on basic expectations. The submission does not meet the minimum requirements.
40-60%	Meets ***basic*** expectations. The minimum requirements are met in whole or in part.
61–74%	Tending to exceed expectations. A ***high-quality*** assignment with additional functionality added.
75-89%	Exceeding expectations and ***very high-quality***. Any mark over 70% requires evidence of ***independent learning*** and cross-fertilization of ideas, i.e. your project must implement a set of features using advanced concepts not covered in depth during the module.
90-100%	***Exemplary***. Your assignment can be used as a teaching aid with little or no editing.