GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computer Science & Applied Physics

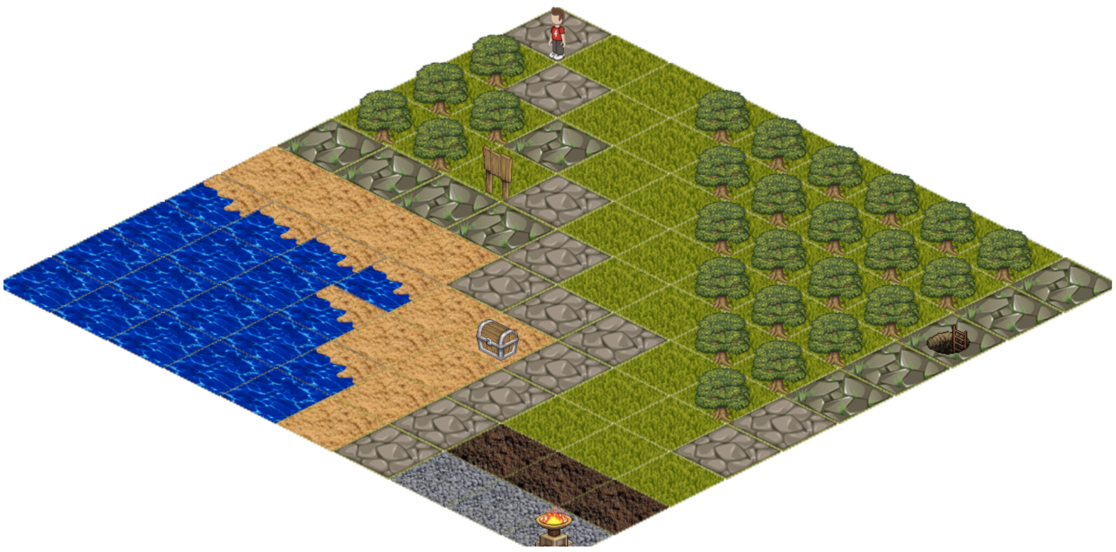
B.Sc. Software Development – Object-Oriented Design Principles & Patterns (2018) **ASSIGNMENT DESCRIPTION & SCHEDULE**

An Isometric 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 order of magnitude. 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 implement 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 the basic features of an isometric game is available on Moodle, but deliberately badly designed. Your objective is to extend, modify and refactor the code provided to create an elegant game design. You should aim to accomplish the following in 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, package and module level.
- Apply creational, structural and behavioural design patterns throughout the game where appropriate. There are obvious uses for factories, builders, flyweights, observers and proxies in an isometric game.
- Use the Swing MVC framework to write custom sprites and game objects.

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 Sunday 6th January 2019***. 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.

- 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 "*An Isometric 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 GMIT student number.
- You must use the **module name gmit.software** and the **package name ie.gmit.sw** for the assignment. Make sure that the module is declared as ***open***.
- The Zip archive should have the following structure (do NOT submit the assignment as an Eclipse project):

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 gmit.software/ie.gmit.sw.Runner** You will need to ensure that the module descriptor contains the declaration **exports ie.gmit.sw** to execute the command above.
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.txt	A text file detailing the main **features** of your application. Marks will only be given for features that are described in the README.
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.gmit.sw** Make sure that you read the JavaDoc tutorial provided on Moodle and comment your source code correctly using the JavaDoc standard.

3. Marking Scheme

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

Element	Marks	Description
Structure	8	The packaging and deployment correct. All JAR, module, package and runner-class names are correct.
README	8	All features and their design rationale are fully documented. The README should clearly document where and why any design patterns have been used.
UML	8	Class diagram correctly shows all the important structures and relationships between types.
JavaDocs	8	All classes are fully commented using the **JavaDoc** standard and generated docs available in the *docs* directory.
Robustness	18	Application executes perfectly, without any manual intervention, using the specified execution requirements.
Cohesion	20	There is very high cohesion between packages, types and methods.
Coupling	20	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. Each of the elements above will be scored using the following criteria:

- 0–30% **Fail:** Not delivering on basic expectations
- 40-59% **Mediocre:** Meets basic expectations
- 60–79% **Good:** Exceeds expectations.
- 80-90% **Excellent:** Demonstrates independent learning.
- 90-100% **Exemplary:** Good enough to be used as a teaching aid

4. Useful Links

- **Web UML Editor:** https://www.draw.io
- **Isometric Tiles Math:** http://clintbellanger.net/articles/isometric_math/
- **Creating Isometric Worlds:** https://gamedevelopment.tutsplus.com/tutorials/creating-isometric-worlds-primer-for-game-developers-updated--cms-28392