IN-COURSE ASSESSMENT (ICA) SPECIFICATION

Module Title: **Software Design Patterns**	Module Leader: **Steven Mead**
	Module Code: **CIS2039-N**
Assignment Titles: **Pocket Beasts**	Deadline Dates: **Element 1: 7th March 2024** **Element 2: 9th May 2024**
	Deadline Time: **4:00pm**
	Submission Method: Online (Blackboard) þ Middlesbrough Tower ¨

Online Submission Notes:

1. Please follow carefully the instructions given on the Assignment Specification

- When Extenuating Circumstances (e.g. extension) has been granted, a fully completed and signed Extenuating Circumstances form must be submitted to the School Reception or emailed to scedt-assessments@tees.ac.uk.

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED INCLUDING MARKING & GRADING CRITERIA

Software Design Patterns (CIS2039-N)

Contents

Introduction

Background “Pocket Beasts” A Deck Building Game

Teesside Toys is a small games company focussing on plastic miniatures. The directors of the company have learned of competing products and the success of trading card games that can span multiple genres. Successful existing examples include “Pokémon TCG”, “Magic: The Gathering”, and online only “Hearthstone” (by Blizzard) which provide lucrative revenues.

Like all exciting new projects, you have been tasked with developing a prototype with a view to bringing an online game to Market. The summer intern (let us call him “Ste”) has developed a simple Java application to get you started (see the attached zip file in the assessments area).

Using the existing “PocketBeasts” code, you will experiment and apply your newfound knowledge of software design patterns and concurrent programming.

Notes:

- You are not required to develop a fully functional card game – the emphasis is the experimentation and application of design patterns and concurrency.
- Errors will be reported when building the existing project because there is an incomplete unit-test for the Card class. Currently, all the tests fail hence the build error. However, the project can still be run.

Element 1 – Research and Design [30%]

You are provided with a skeleton application that, although functional, isn't easily maintainable, extensible and generally lacking in the application of good object-oriented programming principles.

Requirements

Using the existing "PocketBeasts" code, you will experiment and apply your knowledge of software design patterns.

You will analyse the existing codebase, figure out how it works currently and identify weaknesses in its current design (or lack thereof) and implementation.

Deliverables

You will produce a set of slides that will be presented in a 6 to 8-minute video that includes the following:

- **UML class diagrams**
    - Referring to **existing code**, critique the current implementation:
        - Identify and discuss weaknesses in its current design and implementation.
        - Discuss some general initial improvements that can be made to make better use of general object-oriented design principles.
            - Where appropriate, include/identify ***abstract classes*** and ***interfaces***.
    - You may use any appropriate tools to aid the generation of the UML class diagrams.
- **"PocketBeast" Cards**
    - Compare and contrast how the ***Decorator***, ***Façade***, *and* ***Strategy*** patterns discussing their appropriateness to the problem of modifying the behaviour of **cards** during a game. Use UML diagrams to aid your discussion.
    - Choose the pattern you believe is most appropriate, making a clear, concise, and well-argued reason for your chosen pattern.

- You will also discuss why you believe the other patterns are not appropriate and/or best suited to the problem.

NO CODE SHOULD BE PRESENTED OR DISCUSSED FOR ELEMENT 1

Element 2 Implementation [70%]

You will continue the development of "Pocket Beasts", this will be done in three phases as described below.

It is recommended that you keep a technical development diary to your project's progress that includes problems encountered, their solutions, and record resources consulted during problem solving and help develop your understanding.

This will aid discussions from your tutor.

Phases 1 & 2: Unit-testing

You are to provide unit-tests where appropriate for phases 1 & 2. There is an incomplete unit-test for the Card class which can be your starting point. You should create unit-tests for other classes, including ones that you create yourself.

Phase 1: DRYing out the existing codebase

Requirements

In the presentation you are required to identify how the current code base could be improved such as DRYing it out by refactoring, making better use of object-oriented design principles.

In this phase you will implement those improvements.

You must keep a copy of your solution to phase 1.

Deliverables

A Netbeans Java project that includes:

1. The modifications identified to improve the structure and overall implementation of the original codebase.

2. JUnit-tests to comprehensively test the correct operation of the class behaviours. You should endeavour for 100% coverage of all unit-testable classes and expect to get a 100% pass rate.
3. Code should make appropriate use of *Javadoc*, which should be meaningful and complete at the field, method, constructor, class and package level. Variables should be scoped correctly and declared with appropriate access modifiers.

The Netbeans Java project must compile, build and execute in the university Linux labs IT0.11 and IT0.13

Phase 2 – Introduction of suitable design patterns

Requirements

Working on a copy of the code from Element 2/Phase 1 you will implement the chosen design pattern identified as the most appropriate in the presentation from element 1 for the "PocketBeast cards". You may choose to change to one of the other stated patterns if you have a well-argued, well-informed reason to do so.

Furthermore, you identify and implement two other appropriate patterns (see note below) that have been presented during the module to improve the architecture, flexibility, and reduce dependencies between the various components (loose coupling).

Note:

You must demonstrate the use of at least two pattern categories (Creational; Structural; Behavioural). Your chosen patterns should not all be from the same category.

Deliverables

A Netbeans Java project that includes:

1. A modified copy of the project created for Element 1/Phase 1 that demonstrates that clearly demonstrates the use of design patterns.

2. JUnit-tests to comprehensively test the correct operation of the additional classes and their behaviours.
   You should endeavour for 100% coverage of all unit-testable classes and expect to get a 100% pass rate.
3. In addition to the unit-tests, you should provide **test-harness simulation(s)** that demonstrate typical exemplar uses of how the design patterns would be used. The test-harness simulation(s) should not require any user input, but demonstrate realistic usage of the game's features will be expected.

The Netbeans Java project must compile, build and execute in the university Linux labs IT0.11 and IT0.13

Phase 3 – Simple chat application

Requirements

Develop a **simple** Java Swing-based API standalone prototype chat application. Two or more separate instances of the application will be connected via sockets and/or named pipes.

The context is that it would be later integrated into the next version of the “Pocket Beasts” that will have a GUI, rather than be text-based as the current version is.

Deliverables

A Netbeans Java project that includes:

1. A standalone prototype chat application demonstrating communication via sockets and/or named-pipes.

The Netbeans Java project must compile, build and execute in the university Linux labs IT0.11 and IT0.13

Further enhancements

Once the essential requirements have been met, you're welcome to introduce enhanced features for those looking to achieve higher marks. Here is a non-exhaustive list of suggestions:

1. Alternative rules and game modes.
2. Monitoring and reporting of abusive chat between players.
3. Game state monitoring and reporting.
4. Spectator mode.
5. Multiplayer state management.
6. Introduce a GUI.
7. Resilient communication: Handles situations where parts of the communication infrastructure are currently unavailable – for example, research and implement the retry pattern and/or the circuit breaker patterns.

Notes:

1. You're welcome to discuss your ideas and their implementations with your module tutors.
2. These are for students wishing to attain the highest marks, pushing the ICA further.
   However, you're not expected to do all of them.
3. You should provide additional simulation(s) of these features being used.

Report

You will submit a professionally presented report [1000 word count] that reflects upon:

a. Actions undertaken from the design feedback.
b. The completeness of the solution.
c. The advantages and limitations of object-oriented design patterns.

Learning Outcomes

Personal and Transferable Skills

PTS1. Justify the impact of the evidence-based decisions whilst solve programming problems using a variety of computer programming concepts.

PTS2. Document a solution with appropriate reflection and defend in a viva.

Research, Knowledge and Cognitive Skills

RKC1. Develop an understanding of the breadth and depth of software development methods informed by professional practice.

RKC2. Select and use appropriate tools to model/design a software artefact with justifications.

RKC3. Reflect on the completeness of a solution.

Professional Skills

PS1. Analyse, design and construct a solution to a problem using appropriate methods and tools.

Marking Criteria

Element 1 – Presentation [30%]						
Criteria	**80+**	**70-79**	**60-69**	**50-59**	**40-49**	**0-39**
Critique of the current implementation [**10%**].	You have gone beyond the basic ICA requirements for this aspect.	Excellent analysis of the existing implementation and has identified most, if not all of the weaknesses in its design and implementation. Excellent discussion of how objected-oriented principles can be used to improve the current design.	Very good analysis of the existing implementation and has identified majority of the weaknesses in its design and implementation. Good discussion of how objected-oriented principles can be used to improve the current design.	Good analysis of the existing implementation and has identified reasonable number of weaknesses in its design and implementation. A reasonable discussion of how objected-oriented principles can be used to improve the current design, although may not be	Adequate analysis of the existing implementation with weaknesses in its design and implementation. The discussion of how objected-oriented principles can be used to improve the current design is weak, and likely to include a few poor/inaccurate choices.	Little or no attempt made.

		UML class diagram is presented very well, accurate and free of syntax errors.	UML class diagram is presented well, generally accurate and generally free of syntax errors.	entirely convincing. UML class diagram is presented reasonably well, generally accurate and generally free of syntax errors.	UML class diagram is included, although it is likely to lack attention to detail and presentation. Likely to not be an accurate reflection of the current design. It is likely to contain syntax errors.	
Proposal of pattern for the "Pocket Beast" cards [20%]	You have gone beyond the basic ICA requirements for this aspect.	An excellent discussion comparing and contrasting the patterns, including their strengths and weakness with regards to the	A very good discussion comparing and contrasting the patterns, including their strengths and weakness with regards to the	A generally good discussion comparing and contrasting the patterns, including their strengths and weakness with regards to the	An adequate, albeit weak, discussion comparing and contrasting the patterns, including their strengths and weakness with regards to the	Little or no attempt made.

		problem they're being applied to. A compelling argument presented about your selected pattern. There is little room for the audience to question your choice. UML diagrams are informative, accurate, and presented to a very high standard	problem they're being applied to. A reasonably compelling argument presented about your selected pattern. The audience is mostly in agreement, although the maybe some questioning doubts. UML diagrams are informative, mostly accurate, and presented to a high standard.	problem they're being applied to. The audience may not be entirely persuaded that you fully understand the patterns being presented. A reasonable argument presented about your selected pattern. The audience may have a number of concerns at authority of your arguments presented. UML diagrams are reasonably	problem they're being applied to. The audience is not persuaded that you fully understand the patterns being presented. A reasonable argument presented about your selected pattern. The audience may have a number of concerns at authority of your arguments presented. UML diagrams are reasonably informative, the	

				informative, the accuracy is questionable in places, presented to an acceptable standard.	accuracy is questionable in places, presented to an acceptable standard.	

Element 2 - Implementation [70%]						
Criteria	**80+**	**70-79**	**60-69**	**50-59**	**40-49**	**0-39**
Phase 1: DRYing out the existing code base. [15%]	You have gone beyond the basic ICA requirements for this aspect.	An excellent attempt. All problems in existing code have been removed by applying appropriate object-oriented features available in Java.	A very good attempt. A majority of the problems with the existing code have been removed by applying appropriate object-oriented features available in Java.	A good attempt. Some of the problems with the existing code have been removed by applying appropriate object-oriented features available in Java.	An adequate attempt. Very few problems with the existing code have been removed by applying appropriate object-oriented features available in Java.	Little or no attempt made.
Phase 2: Introduction of suitable design patterns	You have gone beyond the basic ICA requirements	An excellent implementation of the chosen design patterns.	A very good implementation of the chosen design pattern. You mostly demonstrate	A good attempt implementation of the chosen design pattern.	An adequate attempt. An attempt has been made to implement the pattern, but you	Little or no attempt made.

[20%]	for this aspect.	It is clear that you without question you understand the patterns, their use and can implement the chosen patterns. Dependencies between collaborating classes in pattern is minimal. Full-featured and extensive simulation of a typical run or run(s) of the chosen scenario.	and understanding of the chosen patterns. Very good simulation. Output/logging is detailed, informative, yet concise feedback to the user about what is happening at each step of the simulation. Consideration of the user by clear formatting and presentation.	There may be some questions about how well you understand the patterns and how to implement them. Dependencies between collaborating classes in pattern are likely to be present. A reasonable simulation, demonstrating most aspects of the chosen	were unable to complete it and/or get it to work as expected. Some attempt at identifying and implementing unit-tests has provided. Superficial attempt to provide a simulation of the of the scenario.	

				scenario and its implementation. Output/logging could be more detailed. Consideration of the user is lacking – information is not presented well.		
Testing [5%] – Unit tests (phases 1 and 2). 100% pass rate of unit-tests is expected.	You have gone beyond the basic ICA requirements for this aspect.	Extensive coverage of unit tests, all appropriate classes have associated unit-test classes. Each unit-test class has a	Very good coverage of unit tests, all appropriate classes have associated unit-test classes, and very good range of test-cases.	Reasonably good coverage of unit-tests, most appropriate classes have associated unit-test classes and a good	Superficial attempt made to provide unit tests.	Little or no attempt made.

		significant number of test cases.		range of test-cases.		
Phase 3: Simple chat application [10%]	You have gone beyond the basic ICA requirements for this aspect.	An excellent attempt exploring and demonstrating a keen understanding of concurrency and issues associated with it. Demonstrates an extensive awareness of the various synchronization features built into the Java language and its APIs.	A very good attempt has been made at exploring and develop a new application using examples that were demonstrated during the module. Generally stable and reliable, although some issues may still be present. Uses Executable and Runnables. Demonstrable/observable behaviour where data sent between different processes via socket **and**	Has made good use of basic synchronization primitives. Some instability of the application caused by concurrency & threading can be observed. Able to demonstrate the use of Thread & Runnable. Stable	Some attempt made to use basic synchronization primitives. The application is generally unstable when concurrency & threading has been introduced. Some attempt made, although not fully realised/implemented. Unstable implementation.	Little or no attempt made.

		ExecutorService, Futures and Callables. Excellent use of an interfaces so that the use of sockets and named pipes as the underlying communication between processes is abstracted from the client code.	named pipes.	implementation Limited to demonstrating data can be successfully sent between 2 (or more) processes via sockets **or** via named pipes.		
Further enhancement [10%]	Additional marks will be awarded to students extending the application further and/or experimenting with other related/relevant projects. A non-exhaustive list of examples provided in the assessment brief. A reasonable attempt must have been made to gain marks here.					Little or no attempt made.
Report [10%]	You have	Excellent	Very good presentation.	Good reflection	Fair presentation.	Little or

	gone beyond the basic ICA requirements for this aspect.	reflection of actions taken from feedback of design. Excellent discussion about the completeness of the solution. Excellent critique of the advantages and limitations of object-oriented design patterns leaving little or no doubt about your understanding of the various	Very good reflection of actions taken from feedback of design. Good discussion about the completeness of the solution. Very good critique of the advantages and limitations of object-oriented design patterns, minor doubts regarding the arguments presented. +/- 10% of the word count.	of actions taken from feedback of design. Good discussion about the completeness of the solution. Good critique of the advantages and limitations of object-oriented design patterns, however, some doubts regarding the arguments presented and/or the use	Adequate reflection of actions taken from feedback of design. Adequate discussion about the completeness of the solution. Adequate critique of the advantages and limitations of object-oriented design patterns, however, may not be convincingly argued or may indicate misunderstanding of the patterns and their use.	no attempt made

		patterns and their applicability.		of the various patterns and their applicability. +/- 15% of the word count.	+/- 20% of the word count.	