COMP 4004A – Fall 2024

About the assignments:

The 3 assignments of this course are concerned with the iterative development and testing of a multi-player game.

The first two assignments involve the text-based implementation of this game on a single computer using a 'hotseat' approach, that is, having each player in turn play *in the same window* using a text-based interface (on a single computer).

Most importantly, players should not see each other's cards and thus, before the game prompts the next player to play, it ends the current player's turn as follows:

1) Asks the current player to leave the hot seat by hitting the <return> key
2) Then 'flushes' the display (i.e., erases all previously displayed information)
3) Identifies who is the next player to play by displaying the id of this player
4) Displays the hand of the next player to play

"Prompting the next player to play" then requires that the game display a context-relevant menu of possible actions/inputs that the player must select from. For simplicity, **for assignment 1, you can assume the input of a player is always a valid one**.

In assignment 3 you are to evolve this game into a web-based version in which different players play in different browser windows on the same computer (in order to be able to record a single video of these players playing the game).

Assignment 1 provides an initial description of the game and of its text-based interface, both of which to be implemented using a specific TDD approach described below. The text-based interface should be **decoupled** from (i.e., easily changeable *without* affecting) the game logic in order to easily change it to a web-based interface in assignment 3.

Assignment 2 adds a few more 'event cards' (see below) and focuses on a systematic approach to acceptance testing, resting on scenario path identification and scenario testing using Cucumber.

Assignment 3 has you evolve your game from its text-based interface to a web interface. Automated testing using Selenium, as well as regression testing, will be the main focus of this 3rd iteration.

Assignment 1 – Due October 15th 2025 at 8AM
Iteration 1 of a game of quests

The objective of this game is for some of its 4 players to accumulate enough *shields* to be knighted. Shields are acquired by successfully completing *quests*. In order to become a knight and win, a player has to accumulate 7 shields. All players who have accumulated 7 or more shields by the completion of the current quest are declared the winner(s) of the game.

The game consists of two decks: an *adventure* deck and an *event* deck. The correct distribution of the cards in each deck must be verified. A discard pile is kept for each of these two decks (i.e., there is an adventure discard pile and an event discard pile). Once a deck runs out of cards, its corresponding discard pile is shuffled and reused as the deck.

The adventure deck consists of 2 types of adventure cards: Foes, and Weapons cards.
There is a total of 100 adventure cards.
Each player initially receives 12 adventure cards. Throughout the game adventure cards are drawn, played and discarded. A player must always keep a *maximum* of 12 adventure cards in their hand after drawing adventure cards (by choosing and discarding excess ones).

A Foe card consists of the letter F followed by a value as follows: 5 (8),10 (7), 15 (8), 20 (7), 25 (7), 30 (4), 35 (4), 40 (2), 50 (2) and 70 (1). The number of Foe cards of each value, in the adventure deck, is given in parentheses after the value. For example, there are 8 F5 cards in the deck with value is 5. There are 50 Foe cards in total.

There are 50 weapon cards (categorized by the letters in parentheses): 6 (D) daggers (value = 5), 12 (H) horses (value = 10), 16 (S) swords (value = 10), 8 (B) battle-axes (value = 15), 6 (L) lances (value = 20) and 2 (E) excaliburs (value = 30). For example, there are 6 “D5” daggers weapon cards.
Displaying the hand of a player means listing foes first in increasing order, then weapons, also in increasing order, with swords before horses.

The event deck consists of 12 Q cards and 5 E cards.
The Q cards are quest cards and have the letter Q followed by a number that ranges from 2 to 5. This number specifies the number of *stages* associated with this quest. So, Q4 denotes a quest with 4 stages. There are a few cards of each type of quest namely: 3 Q2, 4 Q3, 3Q4 and 2 Q5 cards. A quest of *n* stages earns its winner(s) *n* shields.

The E cards denote events:

- There is 1 “Plague” card. The player who draws this card immediately loses 2 shields.
- There are 2 “Queen’s favor” cards. The player who draws this card immediately draws 2 adventure cards.
- There are 2 “Prosperity” cards. All players immediately draw 2 adventure cards.

When an event card is drawn, its event is carried out and the event card is immediately discarded and the turn of the current player ends. When an event card makes a player lose *2* shields, this player ends up with 0 shields if that player has less than *2* shields.

As previously mentioned, the game has exactly 4 players. These players are referred to as P1, P2, P3 and P4. The *order of play* is P1, then P2, then P3, then P4 and then P1 and so on. When the game is started, each player automatically receives 12 random adventure cards from the adventure card deck (which must be correctly updated). The *current player* is the player whose turn it is.

Following the order of play, the current player draws the next *event* card. If that card is an E card, its effect is carried out and then it's the turn of the next player. If the current player draws a Q card, then a *sponsor* for that quest *may* be found and, if so, that sponsor must set up that quest and then that quest must be played out. These steps are explained below. The turn of the current player ends when the current quest has been dealt with. There are no concurrent quests.

When the current player draws a quest card, that player can elect to sponsor that quest or may decline to sponsor it. If that player declines sponsoring the current quest, the next player in the order of play is given the opportunity to sponsor that quest. That player can sponsor or decline. If declined, that quest is offered to the next player and so on. If all players decline to sponsor the current quest, that Q card is discarded and the current player's turn (i.e., the turn of the player who drew that Q card) ends with nothing else happening. Please note that deciding on sponsoring or not a quest does *not* involve drawing additional cards.

When a player decides to sponsor a quest, s/he builds each stage of that quest using the cards currently in their hand. Each stage must consist of a single Foe card and zero or more *non-repeated* Weapon cards (viz. a stage with no foes, or 2 foes or 2 horses is invalid and the game must reject it as illustrated in use case UC-05 shown later in this document). The *value* of a stage is computed as the total of the values of its foe card and weapon card(s). So a stage that consists of F10 + a horse + a lance has a value of (10+10+20) = 40.

Before a quest is resolved, each stage is created as a set of adventure cards. The cards making up each stage of that quest are only visible to the sponsor of the quest. Each stage must be identified, that is, the sponsor must specify which set of cards is for the first stage, which one is for the second and so on. **Most importantly, the value of each stage must be strictly greater than the value of stage that precedes it. This rule must be enforced by the game, signalling an error to the sponsor if it is not respected (see UC-05 below).**

A crucial simplification: A player who decides to sponsor a quest must NECESSARILY have cards that allow for the construction of a valid quest. None of the assignments will deal with a player who sponsors but can't build a valid quest. In other words, never have a player sponsor a quest that they can't build correctly.

Once and only once a quest is built, each player *other* than the sponsor is prompted and must immediately decide if s/he participates in this quest. Then, in order (i.e., from the first to the last stage), each stage of the quest is resolved as follows:

- If there are any participants for that stage, then each of them draws 1 adventure card (then reducing their hand to 12 cards if necessary).
- Each participant for that stage then prepares, in turn, an attack that consists of 1 or more *non-repeated* weapon cards. The value of this attack is the sum of the values of these weapon(s). For example, a dagger, a horse and a lance amounts to an attack of 5+10+20 = 35.
- A participant in the current stage becomes a participant for the next stage of a quest if and only if their attack has a value *equal or greater* than the value of the current stage. If this is the last stage of the quest and a participant's attack is >= to the value of that stage, then that participant has won the quest and earns the number of shields associated with that quest.
- A participant with an attack of value strictly smaller than the value of the current stage *fails* the quest, cannot participate in its subsequent stages (if any) and receives no shields. If all participants of the current stage fail, then the quest has no winner and is abandoned (i.e., the next stages, if any, are not resolved).
- Each participant of a stage, whether s/he wins or loses, must immediately discard all cards used by their attack.

A quest is completed if it has no participants, that is, everyone declines it. Alternatively, a quest is completed once its winner(s) ha(s/ve) been determined or all initial participants have eventually failed this quest (i.e., this quest has no winner).

Once a quest is completed, shields (one per stage) are distributed to each of the winner(s) of the quest (if any). At that point, one or more players may have reached 7 shields and, if so, is/are reported by the game as the winners of the game (which then ends).

Once a quest is completed, *unless* it has led to one or more winners of the game, the sponsor discards **all** the cards used to build the quest and then draws the *same* number of adventure cards + as many additional adventure cards as there were stages in this quest. If necessary, the sponsor then reduces their hand to 12 adventure cards. The sponsor of a quest does not receive shields. After the sponsor of a quest has discarded all cards used for the quest and then drawn new adventure cards, the turn of the current player (i.e., the player who was *initially* offered the sponsorship of the current quest) has ended and the turn of the next player starts, that is, the next player according to the order of play, becomes the current player and plays.

The logic and interface of this game, described in the previous rules, can be captured in the following use cases:

ID: UC-01
Name: 4 players play a game of Quests
Description: 4 players join in a game of quest and take turns until one or more of them win.
Actors: 4 players + the game
Triggering event: The game application is started
Sequence:

1. The game sets up the adventure and event decks
2. The game distributes 12 adventure cards to each player, updating the adventure deck.
3. In the order P1 then P2 then P3 then P4 then back to P1, each player plays a turn (UC-02) **until** one or more winners are identified by the game
4. The game displays the id of the winner(s) and terminates

Post-condition: The game has terminated
Resulting event: The ids of the winner(s) are displayed

ID: UC-02
Name: A player takes their turn
Description: The current player plays their turn.
Actors: 1 player + the game
Triggering event: The game indicates whose turn it is and displays this player's hand
Sequence:

1. The game 'draws' (i.e., displays) the next event card

2a.1 The current player has drawn an E card
2a.2 The game carries out the action(s) triggered by this E card, namely:

i. Plague: current player loses 2 shields
ii. Queen's favor: current player draws 2 adventure cards and possibly trims their hand (UC-03)
iii. Prosperity: All players draw 2 adventure cards and each of them possibly trims their hand (UC-03)

OR
2b The game has drawn a Q card (UC-04)

3. The game indicates the turn of the current player has ended and clears the 'hotseat' display once that player presses the <return> key
4. The game checks if one or more players have won, i.e., has accumulated 7 or more shields

5a The game determines there are one or more winners (enabling step 4 of UC-01)
OR
5b There are no winners and the game indicates it is the turn of the next player (who is assumed to then sit in the 'hotseat') and displays the hand of that player
(i.e., triggers UC-02 for the next player)

ID: UC-03 **Name**: A player trims their hand
Description: The player reduces their hand to 12 cards. **Actors**: 1 player + the game
Triggering event: The player has more than 12 cards in their hand
Sequence:
1. The game computes **n**, the number of cards to discard by that player
For **n** times:
  2.1 The game displays the hand of the player and prompts the player for the position of the next card to delete
  2.2. The player enters a valid position
  2.3. The game deletes the card from the player's hand and displays the trimmed hand

ID: UC-04 **Name**: A Q card is drawn
Description: A quest is played out. **Actors**: 4 players + the game
Sequence:
1.Starting with the *current* player, the game prompts a player as to whether or not that player sponsors the current quest.

2a All players decline to sponsor this quest, in which case the quest has ended, as well as the turn of the current player (and this UC)
OR
2b A sponsor has been found

3. The sponsor sets up a valid quest (UC-05)
4. For each stage of the quest in order:
   - 4.1 The game determines and displays the set of *eligible* participants for that stage
   - 4.2 The game prompts *in turn* each eligible participant as to whether they withdraw from the quest or tackle the current stage of the quest. An eligible participant who withdraws becomes *ineligible* to participate in subsequent stages of this quest.
   - 4.3 An eligible participant who chooses to participate draws 1 adventure card and possibly trims their hand (UC-03)
   - 4.4 The quest ends if there are no participants for the current stage (go to step 5).
   - 4.5 Otherwise each participant for the current stage in turn sets up a *valid* attack (UC-06)
   - 4.6 The game resolves the attack(s) against the current stage:
     - 4.6.1 participants with an attack less than the value of the current stage lose and become ineligible to further participate in this quest.
     - 4.6.2 participants with an attack equal or greater to the value of the current stage are eligible for the next stage (if any). If this is the last stage, they are winners of this quest and earn as many shields as there are stages to this quest.
   - 4.7 All participants of the current stage have *all* the cards they used for their attack of the current stage discarded by the game.
   - 4.8 Unless this is the last stage, the quest ends if there no eligible participants for the next stage. If this is the last stage, the shield total of each winner (if any) is increased and the quest ends.
5. All cards used by the sponsor to build the quest are discarded by the game

Post-condition: The quest has ended

ID: UC-05
Name: Sponsor sets up a **valid** quest
Description: The sponsor sets up the stages of a quest
Actors: 1 player + the game
Sequence:
For each of the stages of the quest:
Until 'quit' is entered **and** the built current stage is valid (ie non-empty **and** of a value greater than the previous stage):

1.The game displays the hand of the sponsor and prompts the sponsor for the position of the next card to include in that stage or 'Quit' (to end building that stage).

2a1 The sponsor enters a **necessarily valid** position of a card (and is re-prompted to do so after an explanation of the invalidity) **until** the selected card is valid (i.e., sole foe or non-repeated weapon card).
2a2 The selected card is included in the set of cards for the current stage, which is displayed
OR
2b 'Quit' is entered *but* the stage has no card associated with it, in which case the game displays 'A stage cannot be empty'
OR
2c 'Quit' is entered *but* the stage is of insufficient value compared to the previous one (if there is a previous stage), in which case the game displays 'Insufficient value for this stage'
OR
2d 'Quit' is entered *and* the stage is valid in which case the cards used for this stage are displayed
Post-condition: The quest is ready to be resolved

ID: UC-06
Name: A participant sets up a **valid** attack
Description: An eligible participant sets up their attack against the current stage
Actors: 1 player + the game
Sequence:
Until 'Quit' is entered **and** the built current attack is valid (ie is a *possibly empty* set of non-repeated weapon cards):
1.The game displays the hand of the player and prompts the participant for the position of the next card to include in the attack or 'quit' (to end building this attack)

2a1 The participant enters the **necessarily valid** position of a card (and is re-prompted to do so after an explanation of the invalidity) **until** this card is valid (ie non-repeated weapon card),
2a2 The selected card is included in the attack, which is displayed
OR
2b 'Quit' is entered, in which case the cards (if any) used for this attack are displayed
Post-condition: The attack of that participant is ready

TDD is about *first* writing tests for some small functional requirement, then the code that makes these tests pass, then refactoring if necessary. In this assignment, we will avoid a purely bottom-up code-driven approach à la Bob Martin, which relies heavily on the intuition and experience of the developer and focuses on *unit* testing. Instead we will adopt a version of TDD centered around Jacobson's notion of *responsibilities*. Beyond TDD, we also require tests for acceptance testing. You are to develop and test this game as follows:

First, from the use cases provided in the previous pages, identify the *responsibilities* of the game. Please note that a step of a use case may involve one or more responsibilities. Also, a single responsibility may be *realized* using one or more methods of some class(es).
For traceability purposes, you are to fill out a table of the form:

RESPONSIBILITY ID	**DESCRIPTION**	**UC STEPS**
RESP-1	Game sets up adventure and event decks	UC-01-1
RESP-2	Game distributes 12 cards from the adventure deck to each player, updating the deck	UC-01-2
RESP-3	At the end of a turn, game determines if one or more players have 7 shields	UC-01-3 and UC-02-5a
RESP-4	Game displays the id of each winner and then terminates	UC-01-4
…	…	…

Second, for each responsibility, you are to:

- Submit a commit labeled **R-TEST** and whose description is a responsibility ID from the previous table. This commit must contain the set of tests used to *validate* this responsibility. Each such test should be named RESP-x-test-y (e.g., RESP-1-test-5) and *should compile*. The set of tests of an R-TEST commit specify the detailed behavior of the single responsibility associated with this R-TEST.
- Then submit a commit labeled **R-CODE** and whose description is the responsibility ID mentioned in the R-TEST commit that immediately precedes this R-CODE commit. In other words, each responsibility is coded and tested using an R-TEST commit immediately followed by its corresponding R-CODE commit. The latter provides all the code required to make the tests of its corresponding R-TEST commit *run and pass*. The R-CODE commit includes the code required to get the game to implement correctly the responsibility at hand.

After one or more pairs of R-TEST/R-CODE commits, it is possible you may want to refactor your code and tests (e.g., to improve your design, to correct mistakes, etc.). A REFAC commit can be used to submit corrections in previously submitted code and tests. REFAC commits should be used sparingly (as a proliferation of them suggests the development process is somewhat chaotic). The description of a REFAC commit should summarized what has been refactored.

Implementing and testing individually the responsibilities of a system is not sufficient testing: it's like testing the components of a car individually, not the car itself. Clearly, sequences of responsibilities must be tested. Use cases were conceived for acceptance testing and are useful for conceptualizing acceptance testing as testing *paths* (aka *scenarios*) through use cases. More specifically, **once** you have implemented all the responsibilities that support a specific path through a use case, you want to (immediately or subsequently) test that path. To do so, you will submit a commit labeled **A-TEST** and whose description is the description of that specific a path through a use case. This commit contains the tests that cover (i.e., *partially* but *sufficiently* exercise) this specific path. For example, one A-TEST commit could address the path 1 – 2a.1.i – 3 of UC-02, which deals with the Plague event card. The description of that A-test commit should capture *in words* the path being tested and give its corresponding sequence of responsibilities. You may want to submit an A-TEST commit as soon as all relevant responsibilities have been tested, or you could want to submit it later in a sequence of A-TEST commits that address a specific use-case.

In theory, acceptance testing would require testing all the possible paths of UC-01 (i.e., all the possible games that could be played!!). This is clearly impossible, which means that acceptance testing is always incomplete and relies on *sufficient coverage* of paths of UCs. But at least one path through each use case (especially UC-01) must be tested.

R-TEST and A-TEST commits pertain to automated testing of the actual game. Consequently, a test in a R-TEST or A-TEST commit **must** call actual methods of the game (as opposed to code specifically develop to make a test pass). In his tutorial, Andrew will explain how to 'rig' an input so that it is set to the specific value a test requires.

We will provide early October a *correction grid* that you will be asked to fill out to tell us how much of the game's functionality you have successfully coded and tested. In this grid, you will also specify the table of responsibilities presented earlier in this document as well as the link to your repository (whose name should be A1-<yourName>-<yourStudentNumber>). Your corrector will access your repo, download your code and tests and run *in one shot* all your tests for responsibilities and scenarios. You will also input in the correction grid a link to a video (created using Zoom) in which you will download and play your game (following a specific script we will provide later), *as well as* show the running in one shot of your tests. A game that cannot be played by itself, or whose responsibility and acceptance tests cannot be run in one shot (i.e., which are not automated), will receive a failing grade.

The correction grid that you fill up must be renamed to A1-<yourLastName>--<yourFirstName>-<<yourStudentNumber> and must be submitted to Brightspace by the deadline.