Team Contract

SYST17796 TEAM PROJECT

Team Name: The Bronze Medalists

Please negotiate, sign, scan and include as the first section in your Deliverable 1.

Please note that if cheating is discovered in a group assignment each member will be charged with a cheating offense regardless of their involvement in the offense. Each member will receive the appropriate sanction based on their individual academic honesty history.

Please ensure that you understand the importance of academic honesty. Each member of the group is responsible to ensure the academic integrity of all of the submitted work, not just their own part. Placing your name on a submission indicates that you take responsibility for its content.

For further information read Academic Honesty Policy on AccessSheridan or visit the faculty office and speak with the Program Support Specialist.

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Omar Musleh	Omar Musleh	991578729
Dhruv Kakadiya		991653956
Kush Patel	Kpatel	991652252
Pizaan Tadiwala	Pizaan	991638701

By signing this contract, we acknowledge having read the Sheridan Academic Honesty Policy as per the link below.

https://policy.sheridanc.on.ca/dotNet/documents/?docid=917&mode=view

Responsibilities of the Project Leader include:

- Assigning tasks to other team members, including self, in a fair and equitable manner.
- Ensuring work is completed with accuracy, completeness and timeliness.
- Planning for task completion to ensure timelines are met
- Any other duties as deemed necessary for project completion

What we will do if . . .

Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member does not deliver component on time due to severe illness or extreme personal problem	Y – OM Y – KP Y – DK Y - PT	a) Team absorbs workload temporarily X b) Team seeks advice from professor __ c) Team shifts target date if possible __ d) Other:
Team member cannot deliver component on time due to lack of ability	Y – OM Y – KP Y – DK Y - PT	a) Team reassigns component __ b) Team helps member X c) Team member must ask professor for reference material __ d) Other:
Team member does not deliver component on time due to lack of effort	Y – OM Y – KP Y – DK Y - PT	a) Team absorbs workload X b) Team "fires" team member by not permitting his/her name on submission __ c) Other:

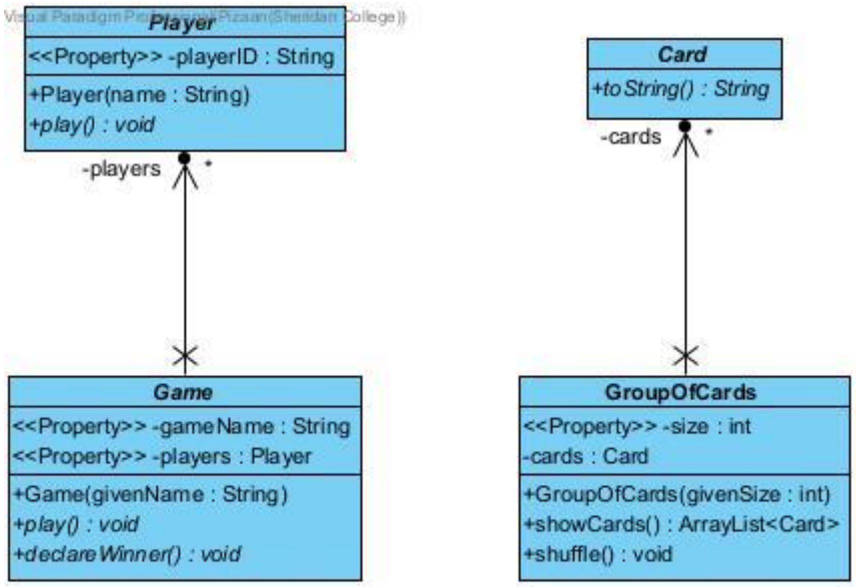
Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member does not attend team meeting	Y – OM Y – KP Y – DK Y - PT	a) Team proceeds without him/her and will assign work to the absent member X b) Team doesn't proceed and records team member's absence __ c) Team proceeds for that meeting but "fires" member after __ occurrences __
A piece of production equipment fails such as a printer, disk drive, or laptop	Y – OM Y – KP Y – DK Y - PT	a) Backup copies will be made and kept in the college __ b) A locker or "share" directory will be used for joint access X c) A photocopy and duplicate disk of all deliverables will be made __ d) Other:
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	Y – OM Y – KP Y – DK Y - PT	a) Team meets and reschedules deliverable X b) Team will cope with constraint __ c) Other:
Team cannot achieve consensus leaving one member feeling "railroaded", "ignored", or "frustrated" with a decision which affects all parties	Y – OM Y – KP Y – DK Y - PT	a) Team agrees to abide by majority vote X b) Team flips coin __ c) Other:
Team members do not share expectations for grade desired	Y – OM Y – KP Y – DK	a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone __

	Y - PT	b) Team votes on each submission's quality X c) Team will ask for individual marking and will identify sections by author __ d) Other:

Scenario	**Accepted Y/N + initial**	**We agree to do the following**
Team member behaves in an unprofessional manner by being rude or uncooperative	Y – OM Y – KP Y – DK Y - PT	a) Team attempts to resolve the issue by airing the problem at team meeting X b) Team requests meeting with professor to problem-solve __ c) Team ignores behaviour __ d) Team agrees to avoid use of all vocabulary inappropriate to the business setting __
Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable	Y – OM Y – KP Y – DK Y - PT	a) Team agrees that this is cheating and is unethical __ b) Friends are friends and should help each other __ c) Team will submit with signature but will advise professor who will take action X
There is a dominant team member who is content to make all decisions on the team's behalf leaving some team members feeling like subordinates rather than equal members	Y – OM Y – KP Y – DK Y - PT	a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote X b) Team will express subordination feelings and attempt to resolve issue __ c) Other:
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted	Y – OM Y – KP Y – DK Y - PT	a) Team forces decision sharing by routinely voting on all issues __ b) Team routinely checks with each other about perceived roles X c) Team discusses the matter at team meeting __

SYST 17796 DELIVERABLE 1
STARTER CODE UML DIAGRAM

Group 1 – The Bronze Medalists (Coup)


Visual Paradigm Professional(Pizaan(Sheridan College))
Player
<<Property>> -playerID : String
+Player(name : String)
+play() : void
-players
*
Game
<<Property>> -gameName : String
<<Property>> -players : Player
+Game(givenName : String)
+play() : void
+declareWinner() : void
Card
+toString() : String
-cards
*
GroupOfCards
<<Property>> -size : int
-cards : Card
+GroupOfCards(givenSize : int)
+showCards() : ArrayList<Card>
+shuffle() : void

SYST 17796 DELIVERABLE 1
DESIGN DOCUMENT TEMPLATE

Group 1 – The Bronze Medalists (Coup)

OVERVIEW

1. Project Background and Description

The purpose of the project is to develop an offline virtual simulation of the card game Coup by Indie Board and Cards, playable by between 3 and 6 players. In the game, each player is given two character cards that are kept secret from the other players. On each turn, players can declare certain effects related to one of the characters to acquire coins (which can be used to declare other effects) or make another player lose influence (discard a card from their hand). Players may bluff if they do not hold the associated character card in their hand. Other players then have the opportunity to challenge (force the active player to reveal if they are bluffing) or block these actions. The ultimate goal is to reduce each player's influence to 0 by forcing them to discard the two cards in their hand. The last player with at least one remaining card is the winner of the round. A detailed description of the game rules can be found at https://www.ultraboardgames.com/coup/game-rules.php.

The project will be developed in Java using core language features in addition to the JavaFX package for GUI development and JUnit for unit testing. Each class of the project is delegated a specific task and is written with high cohesion and loose coupling in mind to ensure functional isolation and ease of extension/modification. Our beta code uses the Model-View-Controller design wherein the representations (model) are hidden from the user. The view (App.java) takes user input which is passed to the controller (Game.java) which then updates the relevant component of the model. Model members are themselves encapsulated as appropriate to ensure updates are made only when and as necessary under meaningful controls.

2. Project Scope

Scope Overview: The aim of the project is to develop a working interactive GUI application which successfully models the card game Coup on a computer supporting play between 3 to 6 players. The project aims to ensure a high degree of usability with an intuitive interface and minimization of software bugs. More detail on the scope of the project can be found in section 4.

Roles & Responsibilities: Roles and responsibilities have been assigned to group members as per the below:

Omar Musleh – Project Leader, Backend Development Primary, Document Approval, Test Design Secondary

Pizaan Tadiwala – QA Lead, Test Design Primary, Backend Development Secondary, Document Review

Dhruv Kakadiya – UI Design Primary, Frontend Development Secondary, Document Drafting

Kush Patel – Frontend Development Primary, UI Design Secondary, Document Drafting

3. High-Level Requirements

Project completion is defined as developing an application which supports the following capabilities:

- Capability to support between 3 and 6 simultaneous players
- Capability to keep track of turns, player's cards, coins and influence
- Capability to determine legal and illegal moves based on coin counts
- Capability to declare effects

- Capability to assess bluffing and challenges, and take the correct corresponding action
- Capability to ensure players are given the appropriate opportunity to challenge or block a declared effect
- Capability to execute declared effect properly if and only if the effect is not successfully blocked or challenged, and execute the effect such that the game responds as expected.
- Capability to determine when a player has been eliminated and prevent them from taking actions
- Capability to ensure that confidentiality of each player's hand is maintained throughout the game
- Capability to play multiple rounds & keep track of how many rounds each player has won
- Capability to show at the game's conclusion the relative rank of each player based on the number of rounds won
- Capability to perform all requirements via an intuitive graphical user interface

4. Implementation Plan

Repository: Code and documents can be found at https://github.com/musleho/syst17796-project-coup.

Development Plan: To ensure that the requirements outlined in section 3 can be met within the specified deadlines, the following milestones have been set.

1. Functional beta code setting up framework and demonstrating proof of concept – COMPLETE
2. Requirements, Use Cases, OOD Description & Complete UML – June 24th
3. Test design & execution including downstream updates to beta code – June 30th
4. UI design completion for release to development – July 8th
5. GUI development and integration with backend – July 29th
6. Final deliverable including tested application – August 5th.

Team members are expected to branch the relevant code to their own development stream and commit code continuously as it is deemed functionally suitable. Code that is deemed reasonably stable by the QA Lead is then incorporated into the 'experimental' branch to be used in testing. Code that has passed all required tests for the given phase of development is then merged into the main branch as a stable version, at the discretion of the QA Lead and Project Leader.

Routine team meetings will be conducted on a weekly basis, taking place each Thursday from 12:00 PM – 1:00 PM. All members are expected to attend weekly meetings unless alternative arrangements have been made prior to the meeting start time. Additional meetings may be scheduled *ad hoc* based on project progress.

Platforms: The application will be developed using Visual Studio Code and IntelliJ IDEA. Tests will be prepared using the JUnit package and extensions for the relevant IDE. UML Diagrams will be prepared using Visual Paradigm and the Draw.io Visual Studio Code extension. Documents will be prepared using Google Docs and Microsoft Word with finalized documents submitted in PDF format. UI will be designed in Adobe XD and developed using the JavaFX package.

5. Design Considerations

As discussed in section 1, our beta code is built using an MVC design pattern making use of several important coding conventions including:

- *High Cohesion:* Each class is given a specific role in the construct of the project such that it can effectively represent the element of the project it is designed for. As an example, our Player class contains a series of properties and methods which only relate to the representation and modification of the player state throughout the course of the game.
- *Loose Coupling:* The dependency of each class in our project on the implementation of other classes is limited to the maximum extent possible. For example, effects which modify player states simply call the relevant Player method with the appropriate parameters in order to execute the effect. The execution of the

effect is entirely independent of how the player state modification occurs, allowing for updates to be made to player methods with no requirement to update the execution of the effects themselves. Similarly, our main program simply calls the execute() method from the relevant effect rather than depending on its specific implementation in order to run properly.

- *Encapsulation:* The vast majority of the game's fields are kept private, and methods which are only intended for use as helpers to other methods in the same class are kept private. An example from our beta code includes the Deck class which has private members activeCards (the cards in the deck) and discardPile (the cards revealed by players). Given the importance of ensuring these fields are only accessed and modified as appropriate (since the count of remaining character cards would influence the outcome of the game), specific methods are used for drawing, returning, discarding or shuffling cards that ensure these critical game elements are not accessed or modified in unexpected ways.