



Crystalcraze Marketplace
Documentation

Created by

Tonkla Wiboonlertwatana 6532076721

Wichayada Chamnansil 6532156221

2110215 Programming Methodology

Semester 1 Year 2023

Chulalongkorn University

CrystalCraze Marketplace

Introduction

The *CrystalCraze Marketplace* is inspired by a board game called Century: Spice Road, in which the players are caravan leaders who travel the famed Silk Road to deliver spices to the far reaches of the continent for fame and glory. Each turn, players perform one of four actions:

- Establish a trade route (by taking a market card)
- Make a trade or harvest spices (by playing a card from hand)
- Fulfill a demand (by meeting a victory point card's requirements and claiming it)
- Rest (by taking back into your hand all of the cards you've played)

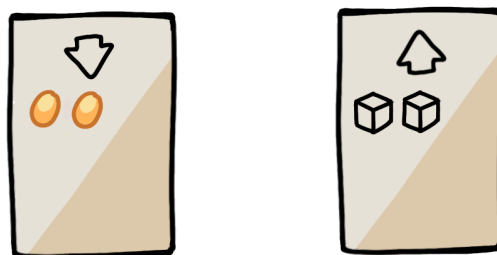
The last round is triggered once a player has claimed their fifth victory point card, and whoever has the most victory points wins.

In our game, we have changed from the spice concept to a gems concept and changed some of the rules.

Game Rules

game ground settings:

1. Each player gets one 'Create 2 yellow' and one 'Upgrade 2' card to form their starting deck, as shown here. (we will talk about this later)



2. In this game, we have a gemstone deal. Each gemstone has a different value, Yellow (Opal) Red (Ruby) Green (Emerald), and white (Diamond) arranged in order.



3. 1st player gets 3 yellow gems in storage
2nd player gets 4 yellow gems in storage
3rd player gets 4 yellow gems in storage
4th player gets 3 yellow and 1 red gems in storage
4. In this game, we have 3 locations: the Gems Shop, the Card Shop, and the Palace.



CrystalCraze Marketplace is played over several rounds. Each player performs one action each turn. When it is your turn, do one of the following actions:

- **Trade gems:** Trade gems from cards in your deck at the gems shop.
- **Acquire:** Get a new trader card at the Card shop.
- **Rest:** Take all the cards you've used back into your deck.
- **Deliver:** Deliver products to customers as they request.

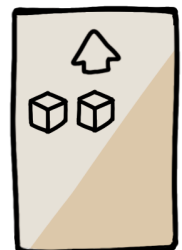
Trade gems

To trade gems, you have to go to the [Gems shop](#) and choose the card from your deck to trade. There are 2 types of trader cards that you can use.



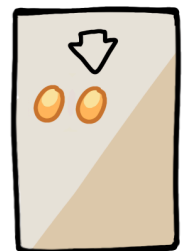
1. Conversion Cards

If you play a conversion card, you may convert gems on your storage into the next most valuable gems (i.e. yellow (opal), red (ruby), green (emerald), white (diamond)) the number of times depicted on the card. For example, 2 opals can convert to ruby or 2 emeralds to 2 diamonds.



2. Exchange Cards

If you play an exchange card, you can trade the above arrow gems from your storage for the gems shown under the arrow. If the card doesn't have any gems above the arrow, it means you can have the gems shown under the arrow by trade with nothing.



Acquire Card

To acquire a trader card you have to go to the [Card shop](#). You have to pay gems of your choice from your storage with the prize above the card you want. After you pay, you will immediately have that card in your deck.

Note: The card on the far left you can acquire for free.



Rest

When you are resting, you take all the cards that you have played before back into your hand. (This allows you to expand your hand throughout the game, i.e. to operate a kind of "deckbuilding"). You can do this action at the [Gems shop](#).



← rest button

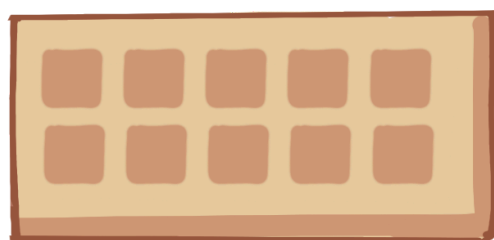
Deliver

To get a score, you have to give the customer the gems they ordered at The Palace. You must have all of the depicted gems in your storage. and give it to the customer by clicking on the check button at the customer. And you will continue to accumulate points by delivering gems like this on and on.



Storage Limit

A storage can keep up to 10 gems. If your actions cause the number of gems to exceed It will be thrown away and cannot be kept.

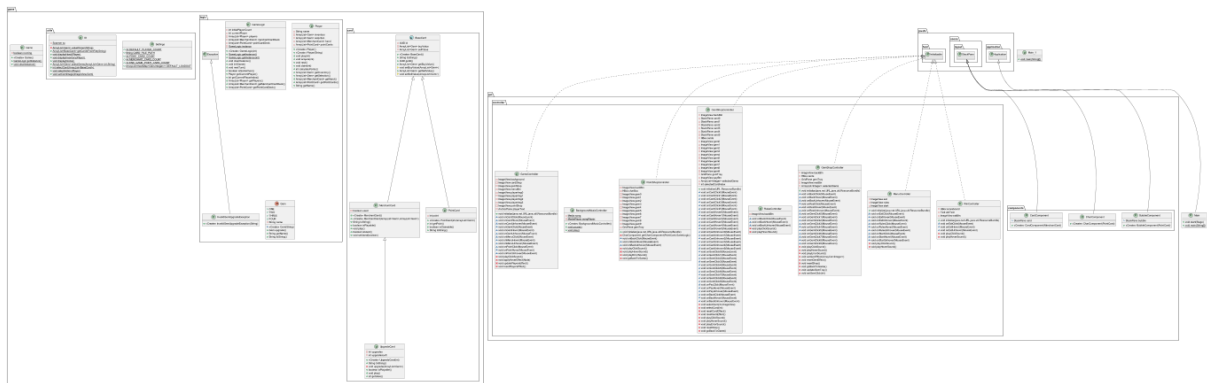


End of the Game

If a player has his fifth customer, the game ends. Each player earns points from their past customer.

All gems better than opal (yellow) earn 1 point per gem. The player with the most points wins the game. In the case of a tie, the player who was later in turn wins.

Class Diagram



1 Package game.card

1.1 abstract class BaseCard

1.1.1 Fields

- final UUID id;	UUID for cards.
- ArrayList<Gem> buyValue	Buy value for each card.
- ArrayList<Gem> sellValue	Sell value for each card.

1.1.2 Constructor

+ BaseCard()	Initialize BaseCase. - Setup UUID
--------------	--------------------------------------

1.1.3 Methods

+ abstract String toString();	String output for CLI play/debug.
+ UUID getId();	Getter for UUID.
+ ArrayList<Gem> getBuyValue()	Getter for buyValue.
# void setBuyValue(ArrayList<Gem> buyValue)	Setter for buyValue.
+ ArrayList<Gem> getSellValue()	Getter for sellValue.
# void setSellValue(ArrayList<Gem> sellValue)	Setter for sellValue.

1.2 class MerchantCard extends BaseCard

1.2.1 Fields

- boolean used	Check whether the card has already been used.
----------------	---

1.2.2 Constructor

+ MerchantCard()	Set the information of the super class.
+ MarchantCard(ArrayList<Gem> buy, ArrayList<Gem> sell)	Set the information of the super class with the given value.

1.2.3 Methods

+ String toString()	String output for CLI play/debug.
+ boolean isPlayable()	Playable state for the card in current turn.
+ void play()	Play the card.

+ boolean isUsed()	get used value
+ void setUsed(boolean used)	set used value

1.3 class PointCard extends BaseCard

1.3.1 Fields

- final int point;	Point valyue of the card
--------------------	--------------------------

1.3.2 Constructor

+ PointCard(int points, ArrayList<Gem> buyValue)	Initialize Pointcard from point value and buyValue.
--	---

1.3.3 Methods

+ int getPoint()	get the value of point
+ boolean isClaimable()	Check whether the card is claimable from shop
+ String toString()	String output for CLI play/debug.

1.4 class UpgradeCard extends MerchantCard

1.4.1 Fields

- final int upgrades;	Number of use for the card.
- int upgradesLeft;	How many upgrades are left.

1.4.2 Constructor

+ UpgradeCard(int uses)	Initializer for the card.
-------------------------	---------------------------

1.4.3 Fields

+ String toString()	String output for CLI play/debug.
- void upgrade(ArrayList<Gem> selectedGems)	Upgrade from selection
+ boolean isPlayable()	Check whether the card is playable.
+ void play()	Play the card in the current turn.
+ int getUses()	Get number of uses left

2 Package game.logic

2.1 class GameLogic

2.1.1 Fields

- final int initailPlayerCount;	Initial number of player
- int currentPlayer;	Current index of player
- final ArrayList<Player> players;	Contains all Player/
- final ArrayList<MerchantCard> merchantCardDeck;	Contain the MerchantCard and UpgradeCard as a deck.
- final ArrayList<PointCard> pointCardDeck;	Contain PointCard as a deck.
- static Gamelogic instance	Game instance

2.1.2 Constructor

+ GameLogic(int initailPlayerCount)	Initialize with player count
-------------------------------------	------------------------------

2.1.3 Methods

+ static GameLogic getInstance()	Get current instance
+ static GameLogic getInstance(int initialPlayerCount)	Get current instance if not exist create new instance with initialPlayerCount
+ void clearInstance()	Clear the current instance (Reset the game)
+ void initGame()	Initialize the game
+ void nextTurn()	Call to change current player
+ boolean isGameOver()	Check if end game condition is met
+ Player getCurrentPlayer()	return the current player
+ int getCurrentPlayerIndex()	return the current player index
+ ArrayList<MerchantCard> getMerchantCardDeck()	return merchant card deck
+ ArrayList<PointCard> getPointCardDeck()	return point card deck

2.2 enum Gem

2.2.1 enum

ONE("OPAL"), TWO("RUBY"), THREE("EMERALD"), FOUR("DIAMOND");

2.2.2 Fields

- final String name;	Store name for each gem
----------------------	-------------------------

2.2.3 Constructor

Gem(String name)	Call gem by its name
------------------	----------------------

2.2.4 Methods

+ Gem upgrade() throws InvalidGemUpgradeException	upgrade the gems for one step.
+ String getName()	get the name of gems
+ String toString()	return name of gems

2.3 class InvalidGemUpgradeException extends Exception

2.3.1 Constructor

+ InvalidGemUpgradeException(String message)	Throw when upgrade is invalid
--	-------------------------------

2.4 class Player

2.4.1 Fields

- final String name;	Store player name
- final ArrayList<Gem> inventory;	Gem inventory for the player
- final ArrayList<Gem> selection	Player selection
- final ArrayList<MerchantCard> hand;	Merchant Card on player's hand
- final ArrayList<PointCard> pointCards;	Store customer/point card

2.4.2 Constructor

+ Player()	Initializer with "Unnamed Player" as name.
+ Player(String name)	Initialize with name.

2.4.3 Methods

+ void play(int cardIndex)	(Player's action) Play card on hand with cardIndex
+ void acquire(int cardIndex)	(Player's action) Acquire card from shop with cardIndex
+ void rest()	(Player's action) Rest card, return cards to usable state
+ void claim(int cardIndex)	(Player's action) Claim point card/customer from the shop
+ int calculatePoints()	Calculate player's point.
+ ArrayList<Gem> getInventory()	Getter for inventory.
+ ArrayList<Gem> getSelection()	Getter for selection.
+ ArrayList<MerchantCard> geHand()	Getter for hand.
+ ArrayList<PointCard> getPointCards()	Getter for pointCards
+ String getName()	Getter for name.

3 Package game.utils

3.1 class IO

3.1.1 Fields

- static final Scanner sc	Scanner for IO and CLI play.
---------------------------	------------------------------

3.1.2 Methods

+ static ArrayList<Gem> valueParser(String valueString)	Parse value from CSV to array of gem
+ static ArrayList<BaseCard> getCardsFromFile(String fileName)	CSV reader for card storage
+ displayHand(Player player)	IO for CLI play (for debugging and testing)
+ static void displayInventory(Player player)	IO for CLI play (for debugging and testing)
+ static void displayDecks	IO for CLI play (for debugging and testing)
+ ArrayList<Gem> selectGems(ArrayList<Gem> inventory, int gemCount, String purpose)	IO for CLI play (for debugging and testing)
+ static int selectCard(ArrayList<BaseCard> cards)	IO for CLI play (for debugging and testing)
+ static void selectAction(Player player)	IO for CLI play (for debugging and testing)
+ static void setGemImage(ImageView imageView, Gem gem)	Set gem image for gems in tray.

3.2 class Settings

Store settings

3.2.1 Fields

- static final int DEFAULT_PLAYER_COUNT	DEFAULT_PLAYER_COUNT = 4;
- static final String CARD_FILE_PATH	CARD_FILE_PATH =

	"src/main/resources/cards.csv"; this file have an information about gems on each card.
- static final int POINT_CARD_COUNT	POINT_CARD_COUNT = 5;
- static final int MERCHANT_CARD_COUNT	MERCHANT_CARD_COUNT = 6;
- static final int END_GAME_POINT_CARD_COUNT	END_GAME_POINT_CARD_COUNT = 5;
+ static final ArrayList<HashMap<Gem,Integer>> DEFAULT_LOADOUT	Default loadout for the gamer

4 Package game

4.1 class Game

4.1.1 Fields

- static boolean running	Check whether the game is running when called by GameController.
--------------------------	---

4.1.2 Constructor

+ Game()	Game initializer
----------	------------------

4.1.3 Methods

+ static GameLogic getInstance()	Get the current instance.
+ static void clearInstance()	Clear instance

5 Package gui.component

5.1 class BubbleComponent extends StackPane

5.1.1 Fields

StackPane bubble;	Bubble pane from FXML file
-------------------	----------------------------

5.1.2 Constructor

+ BubbleComponent(PointCard pointCard) throws IOException	Constructor for from point Card
---	---------------------------------

5.2 class CardComponent extends StackPane

5.2.1 Fields

StackPane card;	Card component from FXML file
-----------------	-------------------------------

5.2.2 Constructor

+ CardComponent(MerchantCard merchantCard) throws IOException	Construct card from merchantCard
---	----------------------------------

5.3 class CharComponent extends StackPane

5.3.1 Constructor

+ CharComponent(PointCard pointCard) throws IOException	Construct char from FXML file
---	-------------------------------

6 Package gui.controller

6.1 class BackgroundMusicController

6.1.1 Fields

- static final Media song	Music to play
---------------------------	---------------

- static final MediaPlayer songPlayer	Music player/
---------------------------------------	---------------

6.1.2 Constructor

+ BackgroundMusicController()	set background music.
-------------------------------	-----------------------

6.1.3 Methods

+ static void pause()	pause the sound.
+ static void play()	play the sound.

6.2 class CardShopController implements Initializable

6.2.1 Fields

- StackPane card0; (card0-card5)	Generated by SceneBuilder for component control
- ImageView backBtn;	
- ImageView gem0; (gem0-gem9)	
- GridPane gemTray;	
- ImageView payBtn;	
- ArrayList<Integer> selectedGems;	
- int selectedCardIndex;	

6.2.2 Methods

+ void initialize(URL url, ResourceBundle resourceBundle)	Initialize the scene
void onCardClick(MouseEvent event)	play click sound and select card.

void onCardHover(MouseEvent event)	play hover sound and make the card glow.
void onCardUnhover(MouseEvent event)	Remove effect from the card
void onGemClick(MouseEvent event)	Select gem to trade
void onPayClick(MouseEvent event)	Check the player validity to trade for the card
- void selectGem(int i, ImageView gem)	Gem selection to trade
- void selectCard(int i)	Card selection
- void resetCardEffect()	Reset effect on cards
- void resetGemEffect()	Reset effect on gems
- void playClickSound()	Play click sound
- void playHoverSound()	Play hover sound
- void playErrorSound()	Play error sound
- void resetShop()	Reset shop before exiting
- void goBackToGame()	Back to main game

6.3 class GameController implements Initializable

6.3.1 Fields

- ImageView background;	Generated by SceneBuilder for component control
- ImageView cardShop;	
- ImageView gemShop;	
- ImageView menuBtn;	
- ImageView playerImg0;	

- ImageView playerImg1;	
- ImageView playerImg2;	
- ImageView playerImg3;	
- ImageView pointShop;	
- AnchorPane playerText;	

6.3.2 Methods

+ void Initialize(java.net.URL,java.util.Resource Bundle)	Initialize the scene
~ void onCardClick(MouseEvent)	Enter card shop
~ void onCardHover(MouseEvent)	Effect for card shop button
~ void onCardUnhover(MouseEvent)	Remove effect from card shop button
~ void onGemClick(MouseEvent)	Enter gem shop
~ void onGemHover(MouseEvent)	Effect for gem button
~ void onGemUnhover(MouseEvent)	Remove effect from card gem button
~ void onMenuClick(MouseEvent)	Return to main menu and clear current instance
~ void onMenuHover(MouseEvent)	Effect for menu button
~ void onMenuUnhover(MouseEvent)	Remove effect from menu button
~ void onPointClick(MouseEvent)	Enter point shop
~ void onPointHover(MouseEvent)	Effect for card point button
~ void onPointUnhover(MouseEvent)	Remove effect from card point button
- void playClickSound()	Play click sound

- void applyHoverEffect(Node)	Apply effect when hovering
- void updatePlayersEffect()	Update current player's effect
- void resetPlayersEffect()	Reset all effect from player

6.4 class GemShopController implements Initializable

6.4.1 Fields

- ImageView backBtn	Back button to return to menu
- HBox cards	Store available card for player
- GridPane gemTray	Player's gem inventory
- ImageView restBtn	Reset button
- ArrayList<Integer> selectedGems	Store player selected gem to upgrade

6.4.2 Methods

+ void initialize(java.net.URL,java.util.Resource Bundle)	Initialize the scene
~ void onBackClick(MouseEvent)	Return to main game menu
~ void onBackHover(MouseEvent)	Apply effect to back button
~ void onBackUnhover(MouseEvent)	Reset back button effect
~ void onRestClick(MouseEvent)	Indicate player rest action
~ void onRestHover(MouseEvent)	Apply effect to rest button
~ void onRestUnhover(MouseEvent)	Reset effect from rest button
~ void onGemClickO(MouseEvent)	Gem selection

~ void onGemClick1(MouseEvent)	
~ void onGemClick2(MouseEvent)	
~ void onGemClick3(MouseEvent)	
~ void onGemClick4(MouseEvent)	
~ void onGemClick5(MouseEvent)	
~ void onGemClick6(MouseEvent)	
~ void onGemClick7(MouseEvent)	
~ void onGemClick8(MouseEvent)	
~ void onGemClick9(MouseEvent)	
- void playClickSound()	Play click sound
- void playHoverSound()	Play hover sound
- void playErrorSound()	Play error sound
- void setGemEffect(ArrayList<Integer>)	Set selected gem effect
- void resetGemEffect()	Reset all gem effect
- void resetShop()	Reset shop to original state
- void goBackToGame()	Go back to main game
- void updateGemTray()	Update the gem tray
- void setGemClick(int)	Set onMouseClicked property for each gem

6.5 class MenuController implements Initializable

6.5.1 Fields

- ImageView exit	Exit button
------------------	-------------

- ImageView rules	Rules button
- ImageView start	Start button

6.5.2 Methods

+ void initialize(java.net.URL, java.util.ResourceBundle)	Initialize the scene
~ void onExitClick(MouseEvent)	Exit the game
~ void onExitHover(MouseEvent)	Set effect for exit button
~ void onExitUnhover(MouseEvent)	Remove effect for exit button
~ void onRulesClick(MouseEvent)	Change to rules page
~ void onRulesHover(MouseEvent)	Set effect for rules button
~ void onRulesUnhover(MouseEvent)	Remove effect for rules button
~ void onStartClick(MouseEvent)	Start the game
~ void onStartHover(MouseEvent)	Set effect for start button
~ void onStartUnhover(MouseEvent)	Remove effect for start button
- void playClickSound()	Play click sound
- void playHoverSound()	Play hover sound

6.6 class PointShopController implements Initializable

6.6.1 Fields

- ImageView backBtn	Generated by SceneBuilder for component control
- HBox charBox	
~ void onExitHover(MouseEvent)	
~ void onExitUnhover(MouseEvent)	

- ImageView gem0	
- ImageView gem1	
- ImageView gem2	
- ImageView gem3	
- ImageView gem4	
- ImageView gem5	
- ImageView gem6	
- ImageView gem7	
- ImageView gem8	
- ImageView gem9	
- GridPane gemTray	

6.6.2 Methods

+ void initialize(java.net.URL, java.util.ResourceBundle)	Initialize the scene
- CharComponent getCharComponent(PointCard,GameLogic)	Setup CharComponents
~ void onBackClick(MouseEvent)	Back button to game menu
~ void onBackHover(MouseEvent)	Set effect for back button
~ void onBackUnhover(MouseEvent)	Remove effect for back button
- void playClickSound()	Play click sound
- void playHoverSound()	Play hover sound
- void playErrorSound()	Play error sound
- void goBackToGame()	Back to main game menu

6.7 class WinController implements Initializable

6.7.1 Fields

- HBox scoreboard	Generated by SceneBuilder for component control
- Text winText	
- ImageView exitBtn	

6.7.2 Methods

+ void initialize(java.net.URL, java.util.ResourceBundle)	Initialize the scene
~ void onExitClick(MouseEvent)	Exit the game
~ void onExitHover(MouseEvent)	Set effect for exit button
~ void onExitUnhover(MouseEvent)	Remove effect for exit button
- void playClickSound()	Play click effect
- void playHoverSound()	Play hover effect

6.8 class RulesController

6.8.1 Fields

- ImageView backBtn	Back button return to main menu
---------------------	---------------------------------

6.8.2 Methods

~ void onBackClick(MouseEvent)	Returns to main menu
--------------------------------	----------------------

~ void onBackHover(MouseEvent)	Set effect for back button
~ void onBackUnhover(MouseEvent)	Remove effect for back button
- void playClickSound()	Play click sound
- void playHoverSound()	Play hover sound

7. Class Main extends Application

7.1 Methods

+ void start(Stage)	JavaFX Application start method override
+ static void main(String[])	Launch application