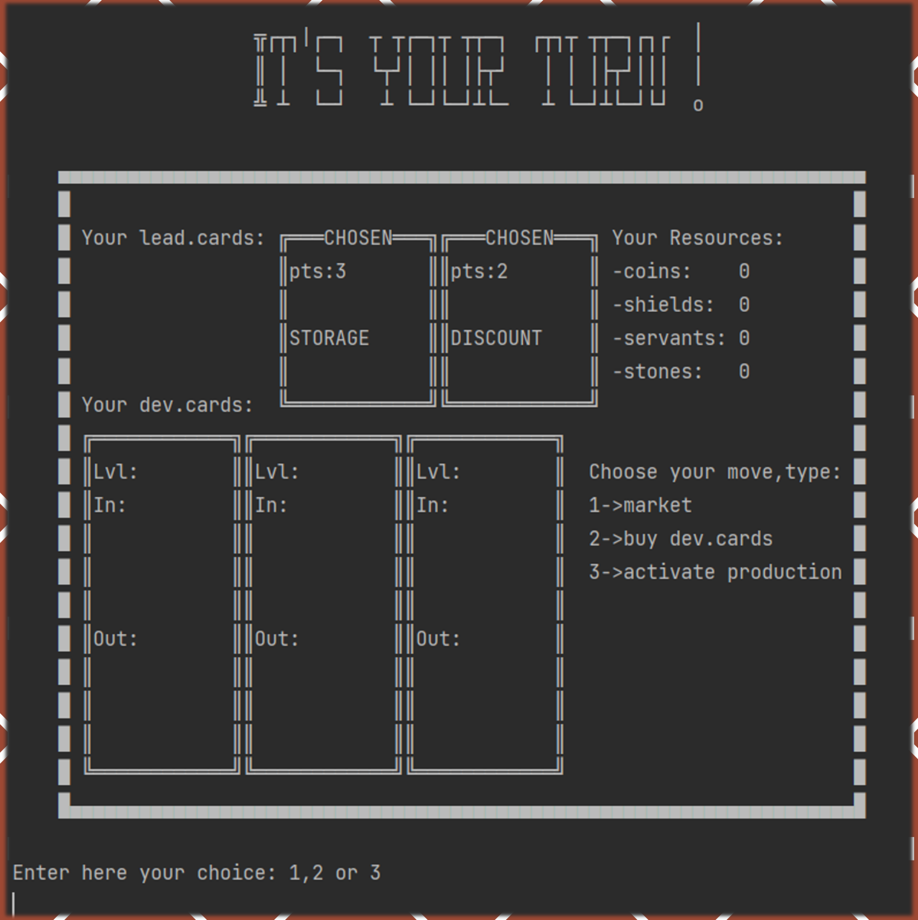


```
IT'S YOUR TURN !

Your lead.cards:  ═CHOSEN═    ═CHOSEN═    Your Resources:
                  pts:3       pts:2       -coins:    0
                                          -shields:  0
                  STORAGE     DISCOUNT    -servants: 0
                                          -stones:   0
Your dev.cards:

 Lvl:        Lvl:        Lvl:          Choose your move,type:
 In:         In:         In:           1->market
                                       2->buy dev.cards
                                       3->activate production

 Out:        Out:        Out:

Enter here your choice: 1,2 or 3
```

Masters of the Renaissance - GC7
It's your turn! Choose your Initial Leader Cards
END TURN
Welcome
Common Board
Your Board
Masters of Renaissance

PROVA FINALE DI INGEGNERIA DEL SOFTWARE

GC07

Francesco Panebianco
Vincenzo Pecorella
Elisa Mariani

CORE DECISIONS

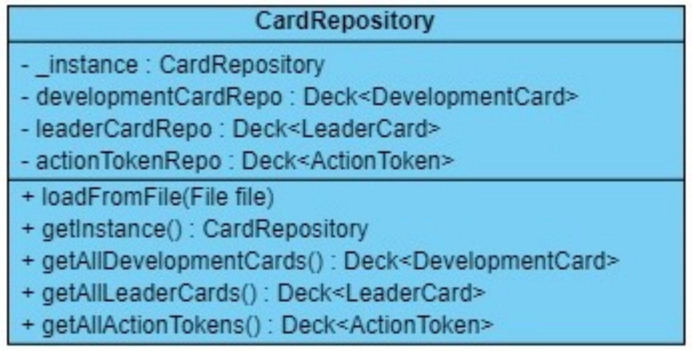
- ASCII Art for Command Line Interface

- **Swing** over JavaFX

- Full Rule Set + 2 Advanced Features (Multiple Game Sessions, Disconnection Resilience)

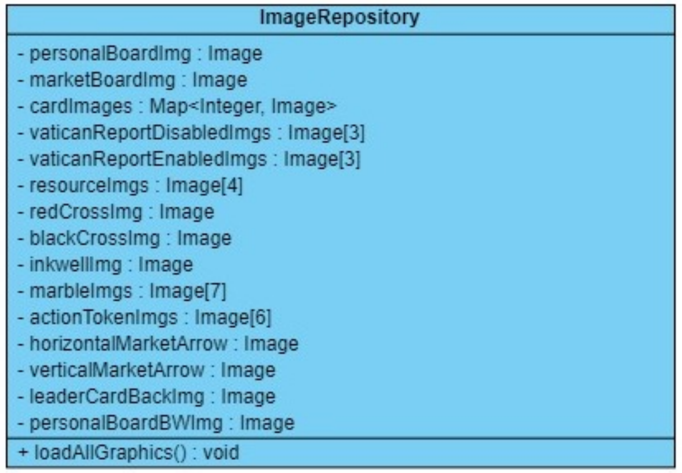
- Thin Client

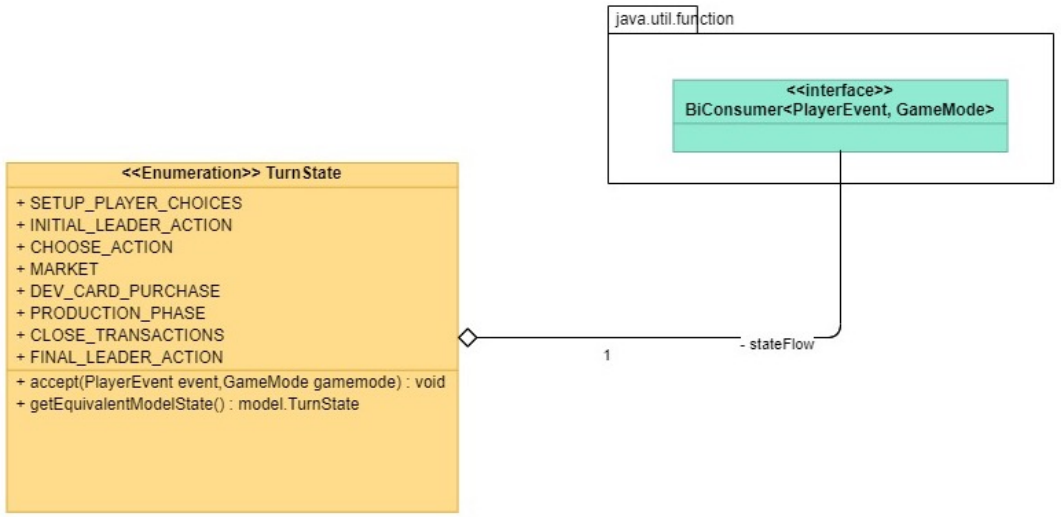
- Client-Server messages and Card Data use JSON serialization (GSON Library)


```
CardRepository
- _instance : CardRepository
- developmentCardRepo : Deck<DevelopmentCard>
- leaderCardRepo : Deck<LeaderCard>
- actionTokenRepo : Deck<ActionToken>
+ loadFromFile(File file)
+ getInstance() : CardRepository
+ getAllDevelopmentCards() : Deck<DevelopmentCard>
+ getAllLeaderCards() : Deck<LeaderCard>
+ getAllActionTokens() : Deck<ActionToken>
```

OTHER DECISIONS

- On Client Side, graphics are loaded once to optimize memory usage and can easily be accessed through a ImageRepository (Singleton


ImageRepository
- personalBoardImg : Image
- marketBoardImg : Image
- cardImages : Map<Integer, Image>
- vaticanReportDisabledImgs : Image[3]
- vaticanReportEnabledImgs : Image[3]
- resourceImgs : Image[4]
- redCrossImg : Image
- blackCrossImg : Image
- inkwellImg : Image
- marbleImgs : Image[7]
- actionTokenImgs : Image[6]
- horizontalMarketArrow : Image
- verticalMarketArrow : Image
- leaderCardBackImg : Image
- personalBoardBWImg : Image
+ loadAllGraphics() : void



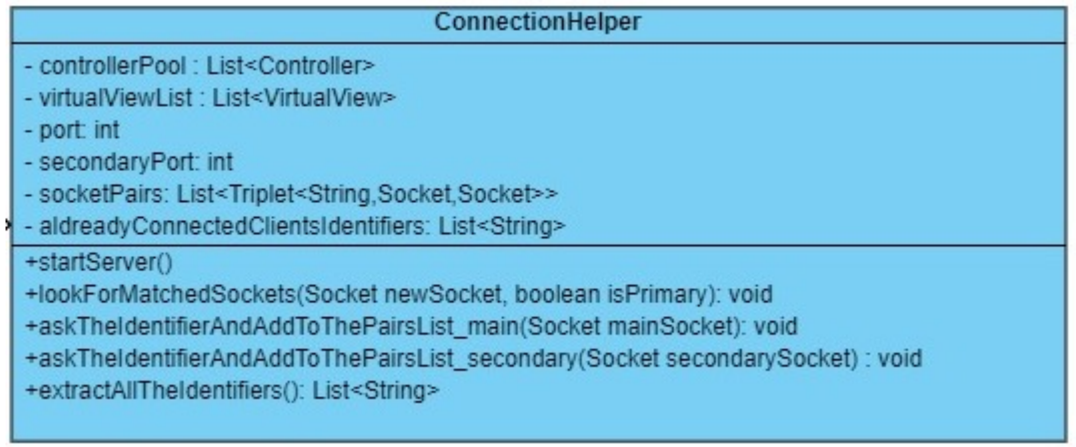
- When the controller receives a PlayerEvent, the message is handled by a BiConsumer encapsulated in each state the player turn may be (TurnState). Each consumer function calls methods from *GameMode*, a controller class containing methods to perform actions on the game session.

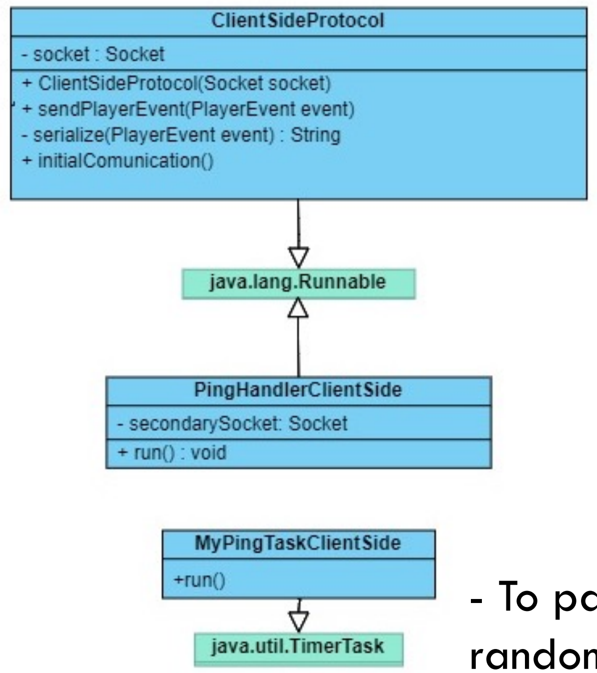
A CRITICAL DECISION

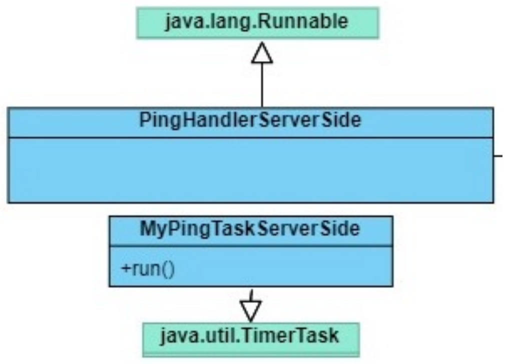
- We decided to implement the FA: Client Reconnection using a **ping mechanism**
- Many reasons led us to this decision:

1. The clients' connection isn't reliable: in some cases they disconnect without notify the server
2. Since the disconnected players skip their turn, by a tactical prospective can be useful to always know who's connected and who's not
3. A simple idea to know when a client is disconnected is to put a timer on player's moves: it's easy to implement but
    - Puts a constraint on users' move time
    - To recognise that a player is truly disconnected a long timeout must pass and all the players have to wait

A CRITICAL DECISION


ConnectionHelper
- controllerPool : List<Controller>
- virtualViewList : List<VirtualView>
- port: int
- secondaryPort: int
- socketPairs: List<Triplet<String,Socket,Socket>>
- aldreadyConnectedClientsIdentifiers: List<String>
+startServer()
+lookForMatchedSockets(Socket newSocket, boolean isPrimary): void
+askTheIdentifierAndAddToThePairsList_main(Socket mainSocket): void
+askTheIdentifierAndAddToThePairsList_secondary(Socket secondarySocket) : void
+extractAllTheIdentifiers(): List<String>





- We then chose to implement a ping mechanism. Handling disconnections properly, especially in the CLI, ended up requiring two sockets: one for the CompressedModel / PlayerEvent messages and one for the continuous *ping* message. With the current implementation, it takes 5 seconds without a ping to disconnect the player.

- To pair the two sockets without ambiguity, a unique identifier for the client is generated randomly and sent upon first connection.

- Thanks to this approach, the thread that receives the message also handles the turn and sends the answer back. This eliminates many syncronization hazards.

CLI & GUI

- **Common Board**

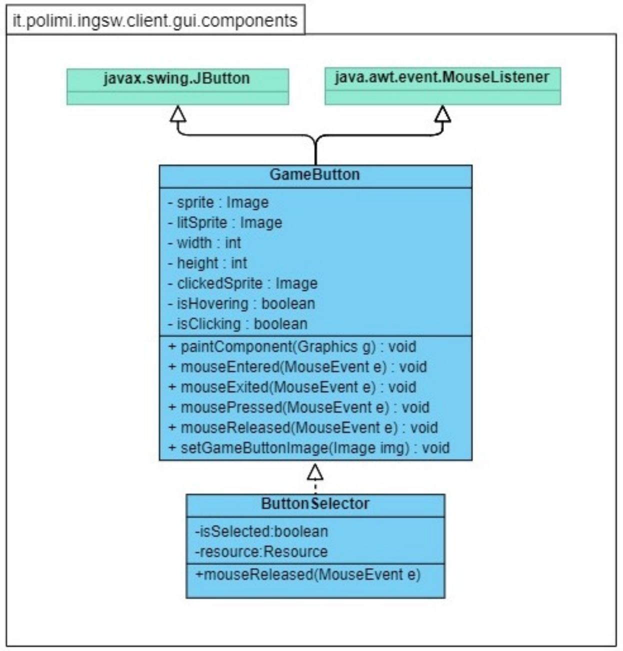
Contains common spaces available to all players: *Market Board* and *Development Card Grid*.

- **Player Board**

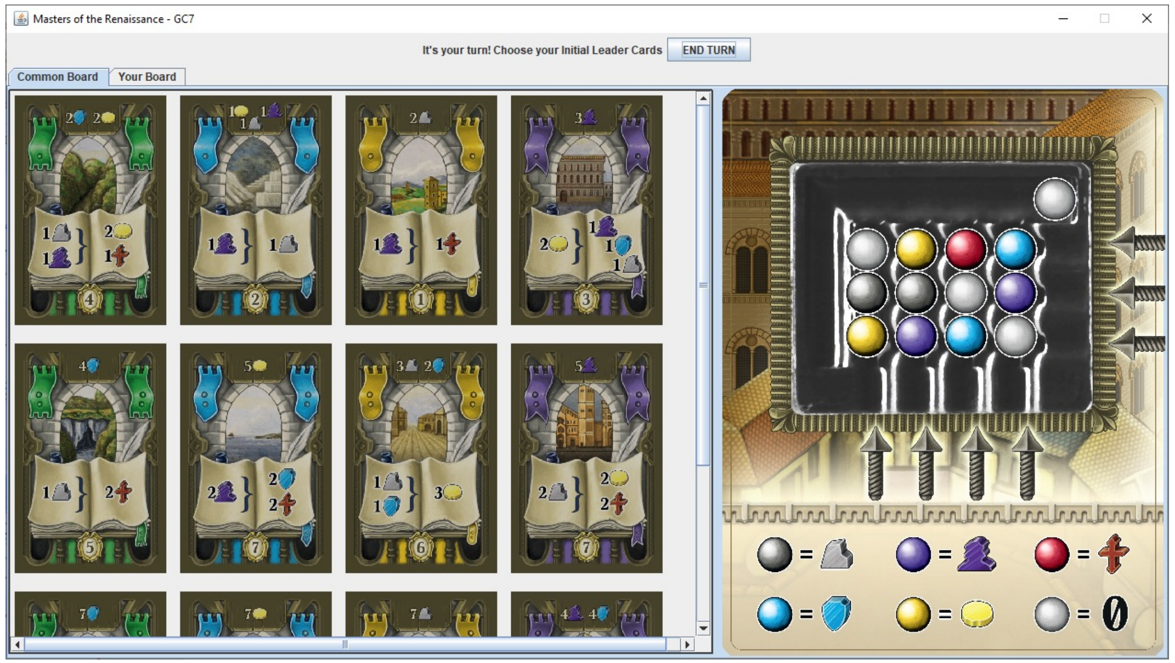
Contains the player specific buttons and views.

- Buttons are disabled for other players' boards.
- Other players' leader cards become visible upon activation

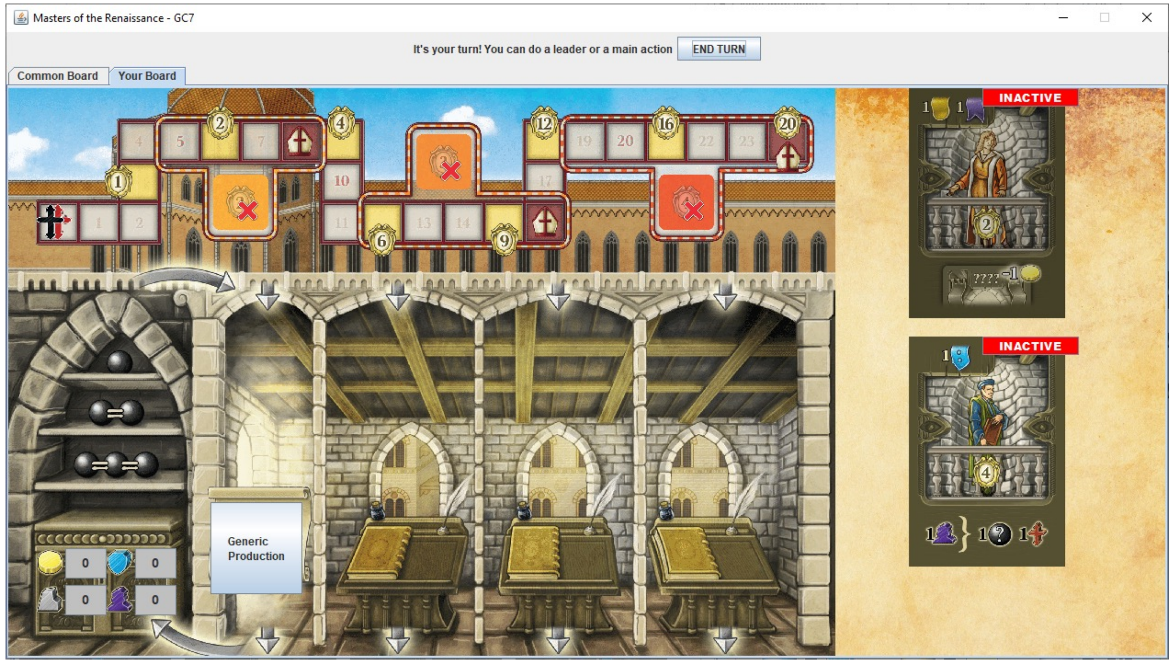
- Selection of cards and market rows / columns is done with *GameButtons*, which have transparent backgrounds and are highlighted when the mouse hovers on them.



it.polimi.ingsw.client.gui.components
javax.swing.JButton
java.awt.event.MouseListener
GameButton
- sprite : Image
- litSprite : Image
- width : int
- height : int
- clickedSprite : Image
- isHovering : boolean
- isClicking : boolean
+ paintComponent(Graphics g) : void
+ mouseEntered(MouseEvent e) : void
+ mouseExited(MouseEvent e) : void
+ mousePressed(MouseEvent e) : void
+ mouseReleased(MouseEvent e) : void
+ setGameButtonImage(Image img) : void
ButtonSelector
-isSelected:boolean
-resource:Resource
+mouseReleased(MouseEvent e)


Masters of the Renaissance - GC7

It's your turn! Choose your Initial Leader Cards


Masters of the Renaissance - GC7

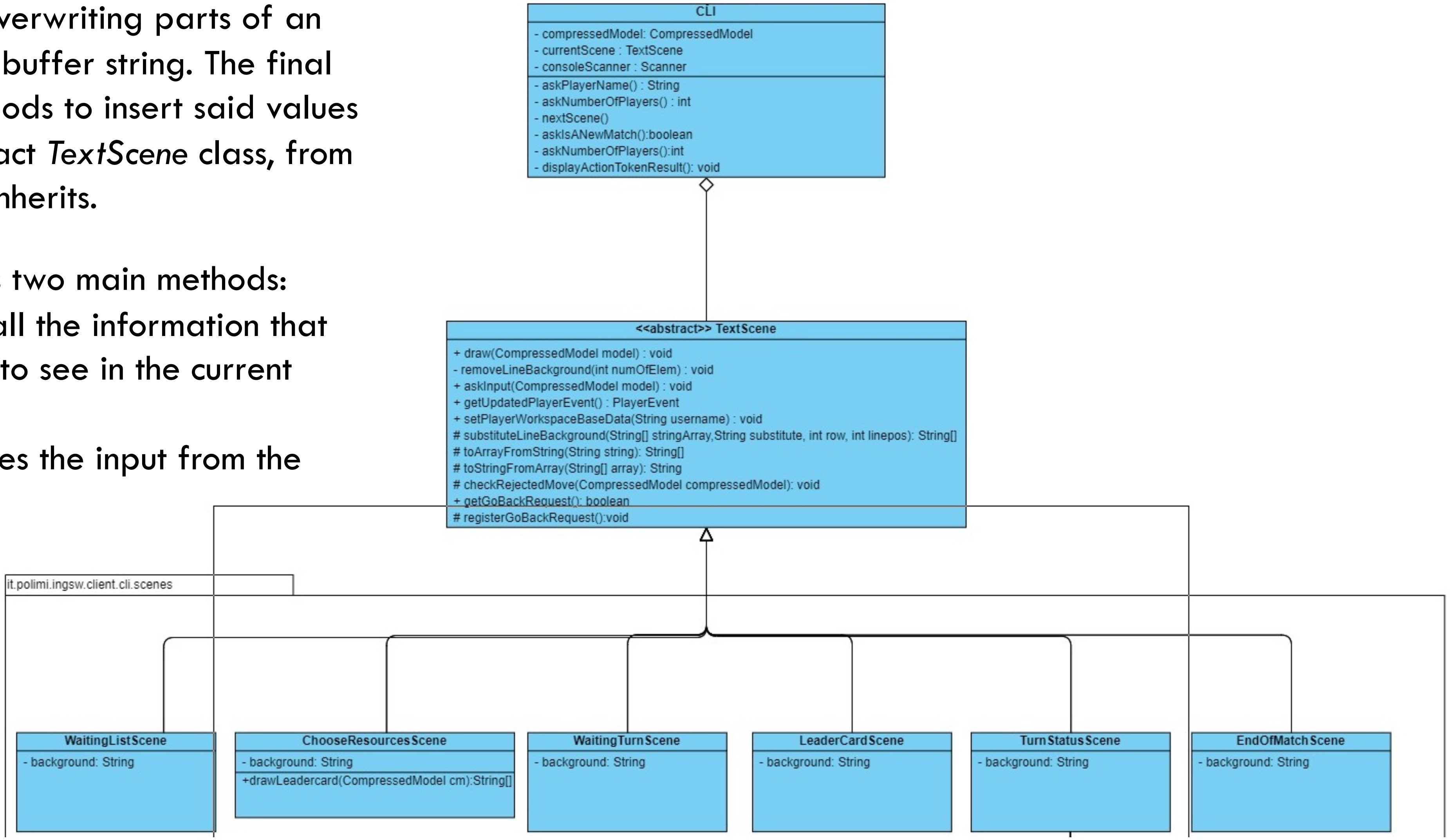
It's your turn! You can do a leader or a main action

CLI & GUI

- Values are inserted by overwriting parts of an ASCII background using a buffer string. The final string is then printed. Methods to insert said values are contained in the abstract *TextScene* class, from which every actual scene inherits.

- The TextScene class has two main methods:
  1. Draw() : it prints all the information that the player needs to see in the current scene
  2. AskInput(): retrieves the input from the player

- The correct scene is shown based on the *TurnState* received from server.

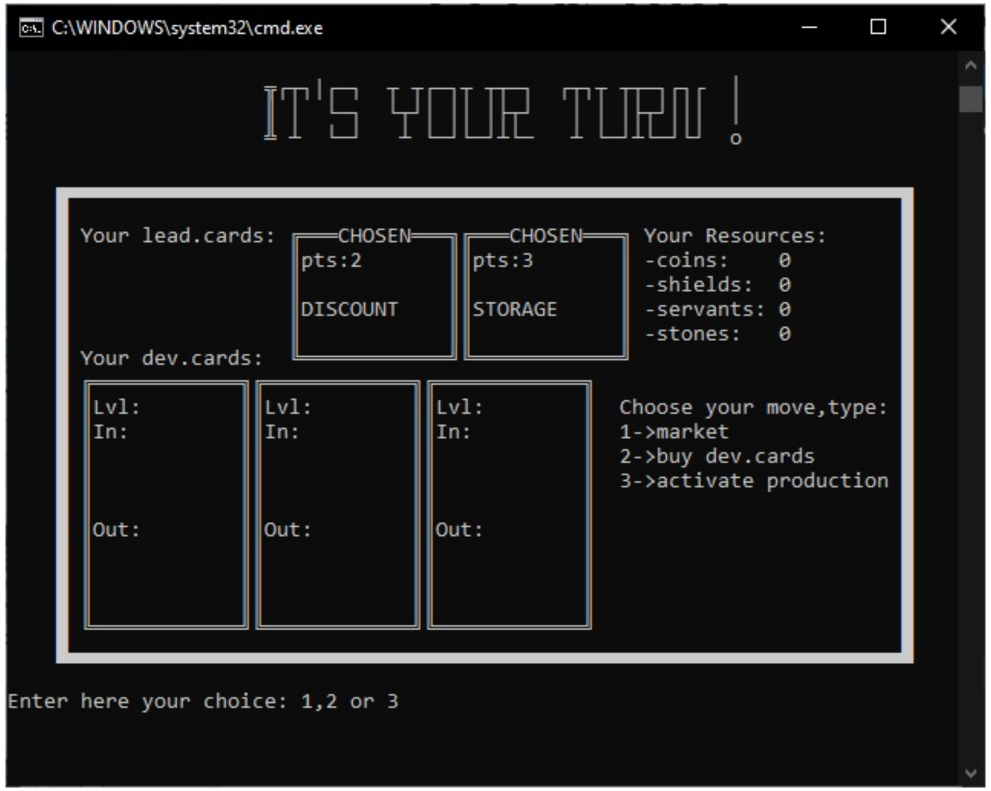

CLI
- compressedModel: CompressedModel
- currentScene : TextScene
- consoleScanner : Scanner
- askPlayerName() : String
- askNumberOfPlayers() : int
- nextScene()
- askIsANewMatch():boolean
- askNumberOfPlayers():int
- displayActionTokenResult(): void
<<abstract>> TextScene
+ draw(CompressedModel model) : void
- removeLineBackground(int numOfElem) : void
+ askInput(CompressedModel model) : void
+ getUpdatedPlayerEvent() : PlayerEvent
+ setPlayerWorkspaceBaseData(String username) : void
# substituteLineBackground(String[] stringArray,String substitute, int row, int linepos): String[]
# toArrayFromString(String string): String[]
# toStringFromArray(String[] array): String
# checkRejectedMove(CompressedModel compressedModel): void
+ getGoBackRequest(): boolean
# registerGoBackRequest():void
it.polimi.ingsw.client.cli.scenes
WaitingListScene
- background: String
ChooseResourcesScene
- background: String
+drawLeadercard(CompressedModel cm):String[]
WaitingTurnScene
- background: String
LeaderCardScene
- background: String
TurnStatusScene
- background: String
EndOfMatchScene
- background: String

CLI & GUI

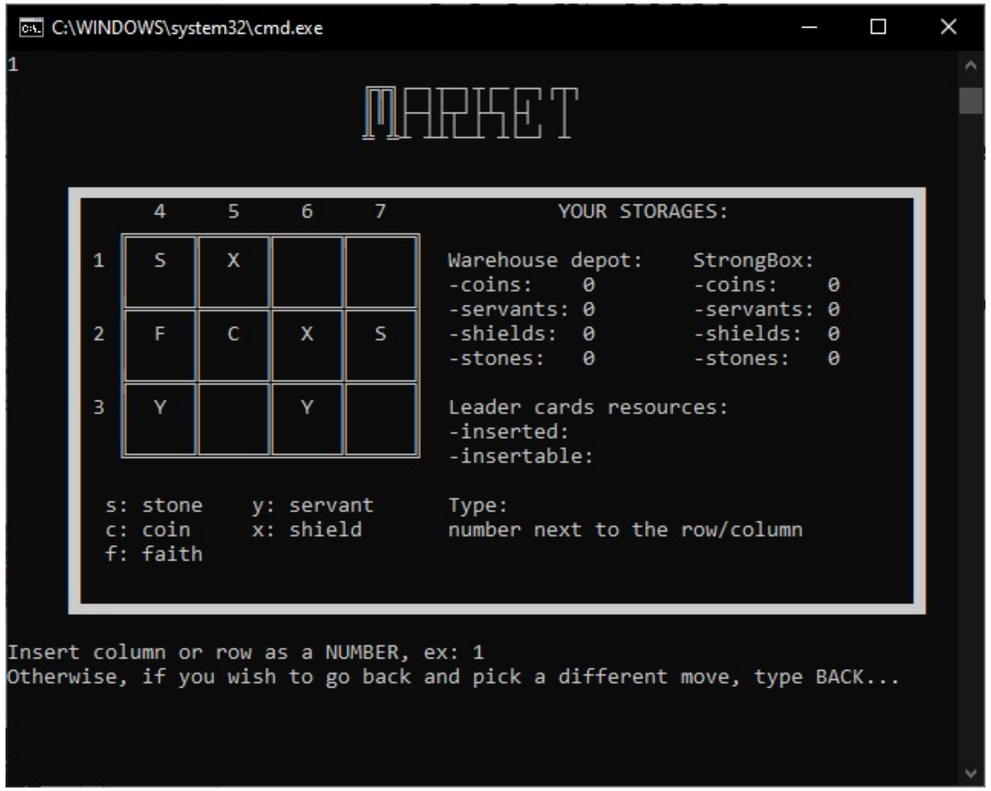
- **CLI - TurnStatusScene**

Here the players can see their Development cards and owned resources.

- They are given the choice to do a *Market Action, Buy a Dev Card* or *Activate Production Powers*



```
C:\WINDOWS\system32\cmd.exe

              IT'S YOUR TURN !

  Your lead.cards:  ══CHOSEN══   ══CHOSEN══   Your Resources:
                    pts:2        pts:3        -coins:    0
                                              -shields:  0
                    DISCOUNT     STORAGE      -servants: 0
                                              -stones:   0
  Your dev.cards:
  Lvl:        Lvl:        Lvl:              Choose your move,type:
  In:         In:         In:               1->market
                                            2->buy dev.cards
                                            3->activate production



  Out:        Out:        Out:



Enter here your choice: 1,2 or 3
```



```
C:\WINDOWS\system32\cmd.exe
1
                    MARKET

        4     5     6     7              YOUR STORAGES:
   1    S     X                   Warehouse depot:    StrongBox:
                                  -coins:    0        -coins:    0
                                  -servants: 0        -servants: 0
   2    F     C     X     S       -shields:  0        -shields:  0
                                  -stones:   0        -stones:   0

   3    Y           Y             Leader cards resources:
                                  -inserted:
                                  -insertable:

    s: stone     y: servant       Type:
    c: coin      x: shield        number next to the row/column
    f: faith


Insert column or row as a NUMBER, ex: 1
Otherwise, if you wish to go back and pick a different move, type BACK...
```