**OOP LAB REPORT**

**Group 16**

**Topic: Ô Ăn Quan**

1. Assignments of members
2. Đinh Ngọc Lập Thành (Trưởng nhóm)

* Schedule and design working timeline.
* Design General and Detail Class Diagram.
* Design and implement UX/UI
  + Package: sourcecode/Frame
  + **CLAIM:** Code uses ideas and snippet from Stack Overflow but doesn’t directly copy or copy with modify from any knowable source.
* Integrate backend code (Tú) into UX/UI
* Write mini project report.

1. Nguyễn Minh Tú

* Design Use Case Diagram.
* Implement player and game mechanics.
* Design presentation slides.

1. Mini-Project Description
2. Mini-Project Requirements

* A detail implementation of O An Quan mechanics and player actions in terminal
* A main screen that enables players to play, see game rules or exit.
* A game screen that allows players to play the game.
* An end game screen that displays who’s the winner and players’ points.

1. Use case diagram:

A diagram of a diagram

Description automatically generated

* Explanation:
  + In the main screen, the user can choose the start playing the game, open the help window to see the game’s rules or exit the game.
  + When click play, the software will create a match and update two player info (name and point)
  + The user can then spread gems, and at any point, choose to exit the match.
  + The software each turn will show current turn and show a timer for that player turn. Finally, it will calculate points and declare a winner.

1. Design

* General class diagram:
* Explanation for the design:
  + Firstly, class main creates a new MainFrame instance called frame that inherited from JFrame and implemented ActionListener.
  + This frame window is the main screen of the game, allows user to choose to play the game by clicking on the startButton, see the help window by clicking on the help Button or choose to exit the game by clicking on exitButton (which will require a second confirmation from the user per the mini-project specification, the main screen window will be dispose if user choose yes).
  + If a user chooses to play the game, the main screen window will be dispose and the MainFrame instance will create a GameFrame instance called gameframe that is too inherit from JFrame and implements MouseListener, thereby create and initialize the game and the game window.
  + The gameframe have a Border Layout, will player 2 information container in the North, player 1 information container in the South and the GameBoard in the center, the player 2 container will also have a count down timer and a notification of how many gems a player has in hand to spread.
  + The GameBoard is implemented as an array of MyPanel instances to align with the backend code and easy retrieval of information.
    - Class MyPanel inherits from JPanel and receives information during GameBoard creation about its shape (square or half-circles) and whether it is in upper row or lower row as well as its index in the array.
    - To indicate number of gems inside a tile, class MyPanel will override paintCompenent method to draw it shape and customize its coordinate according to position.
    - The color pallete of tile will change according to whether it is in static, indicating gem spreading or indicating gem collecting.
    - Method drawGems also utilize paint Component to constantly paint the number of gems inside a tile.
    - Method setGemsIndicator will also be called each repaint interval to update in number how many gems are in a tile.
  + By default, player 1 will be the first to go. Each turn is counted down by timerCountDown method from 60 seconds.
  + timerCountDown will first check if both Quan are eaten or not, if yes it will end the game and began calculating to determine the winner.
  + If not, then it will signal the current turn and begin count down from 60 seconds. If the current turn’s player does not make a move after 60 seconds, it will automatically change turn and begin counting.
  + A player in turn can only move tile that has gems > 0, is a center (square) tile and in their respective row.
  + When a player clicks on a direction arrow to begin spreading, the timer will stop, the gameframe will be disabled to prevent further player action and we called spreadGems method at the tile’s index and provide the direction.
  + If spreadGems take the index of a half circle, it will return immediately, but if not, then it will take that tile’s gem and began spreading.
  + After spreading, method recursiveSpreadGems will receive the starting index and starting direction, this combine with current index will determine the which direction we should spread gems again if the next tile has gems > 0.
  + If next tile gem == 0, we call addPoint method to current player and then finish the turn by enable action from player to frame again.
  + From the second turn onward, afterTurnAction will be called at the end of each turn to check if the upper and lower row has any gems, if not then the player responsible for it will be deducted 5 points to fill each tile with 1 gem.