**OOP LAB REPORT**

**Group 16**

**Topic: Ô Ăn Quan**

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

* Design detailed Class Diagram and Use Case Diagram.
* Design GUI and implement game logic.
  + Package: sourcecode/Frame
  + **CLAIM:** Code uses ideas and snippets from Stack Overflow but doesn’t directly copy or copy with modify from any knowable source.
* Write mini project report.
* Design project slide.

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

* A main screen that enables players to play, show guide or exit.
* A game screen that allows players to play the game, including making moves to spread gem across the gameboard, each turn showing clear which player’s turn it is.
* An end game screen that displays who’s the winner and players’ points.

1. Use case diagram:

A diagram of a computer

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

* Class diagram:

A yellow paper with black text

Description automatically generated

* 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 helpButton 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 then be disposed of after the two verifications are met.
  + If a user chooses to play the game, the main screen window will be disposed and the MainFrame instance will create a GameFrame instance that inherited from JFrame and implements MouseListener, thereby create and initialize the game and the game window.
  + The game frame has a Border Layout, with player 2 information’s container in the North, player 1 information’s container in the South and the GameBoard in the center, the player 2 container will also have a countdown 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.
    - 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 firstly 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. If the current turn’s player does not make a move after 60 seconds, it will automatically change turn.
  + A player in turn can only move tile that has the number of gems above 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 the number of gems above 0.
  + If the next tile’s 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/or 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.