Here's a example of a proposal ***(using Swing, not HSA2)***

Note: items in red have not been completely worked out yet.

Program Name: **Pente / Go-moku**

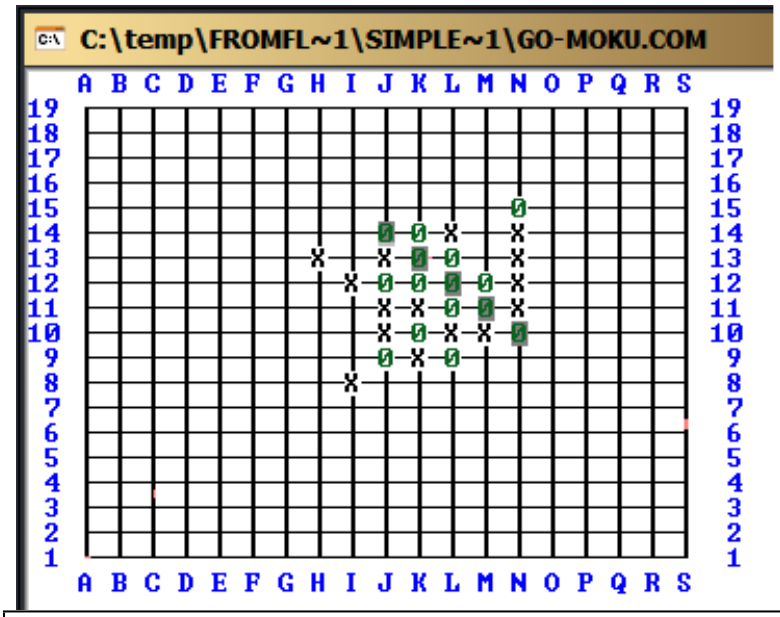
Problem Definition: play TicTacToe on an 13x13 board. First player to get 5 in a row wins.

Description: The program will start out allowing the user to play against another person or against the computer. Initially the computer AI will just choose random positions. Later on, as time permits, I'll improve the algorithm to use intelligence.

Rules: There are two colors representing each player. Players alternate turns, placing a token somewhere on the board. Tokens are placed inside each square. Tokens cannot be moved or removed during a game. The first player to get five in a row, horizontally, vertically, or diagonally, wins.
The board size can be changed in the options from 13x13 to 19x19.
There is also a capture rule that may be implemented later.

Main Screen

- Diagram of main screen is below.
  Note: this is from a DOS version, so it won't be exactly like this. There will be a menu at the top. I don't see the necessity of labeling rows and columns, so I won't do that either.



- There will also be a other screens that pop up as needed (e.g. game options, rules)
- Menu layout:
  - Game:
    - New, Options, Load, Save, Quit
  - Help
    - Rules, About

Design

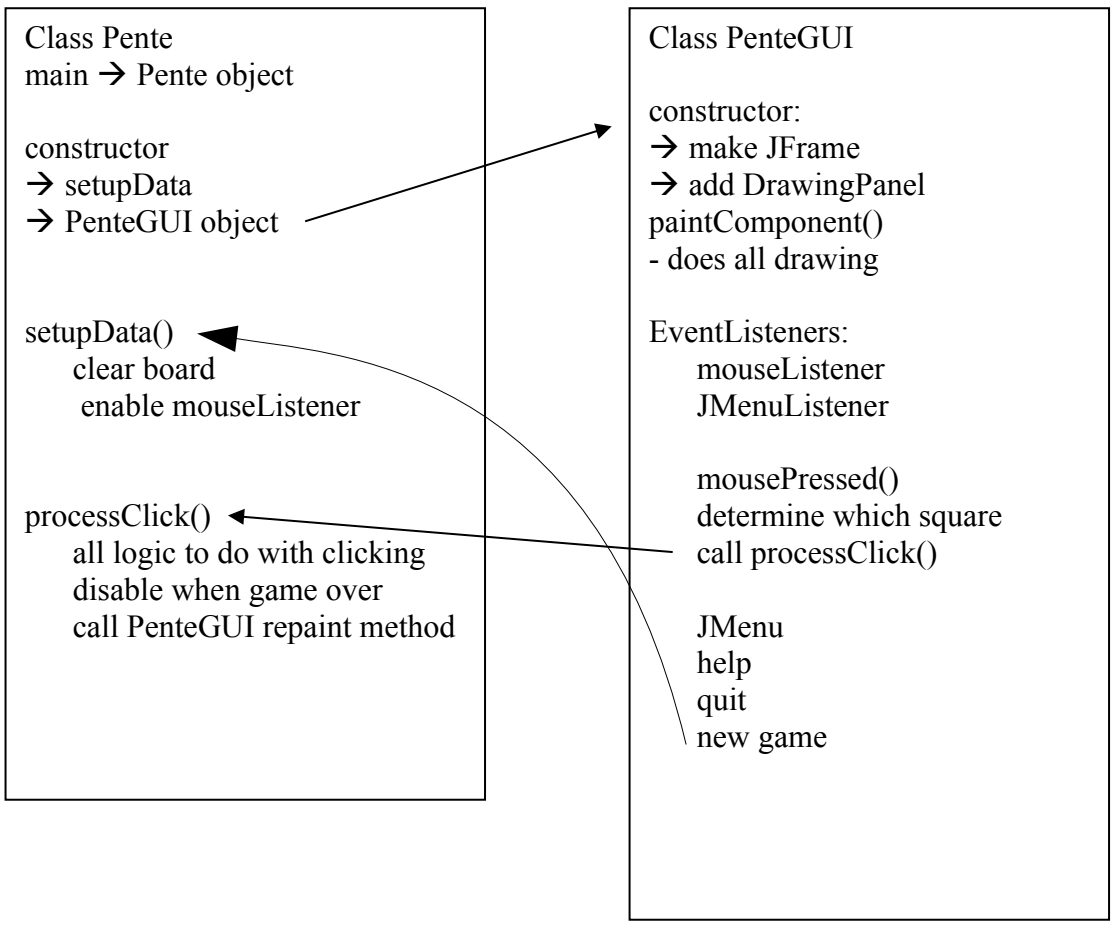
- The overall intent of my design is to separate the graphics from the calculation part of the game, which is always a good way to do things. *For a simple game, it wouldn't be unrealistic to put both parts into one file. It is only really significant when the calculation part gets more complex (e.g. with AI).*

- I/O
  - the input while playing the game will all be done by clicking on the screen.
  - There are two places where one can click:
    - the menu system to select various choices, start game, quit, save game, etc.
    - the board to select where to place your token.
  - the output is only to the display.
  - the graphics will all be drawn using lines and circles, not custom made objects nor Swing objects (e.g. JButtons).
  - Perhaps, later, if games can be saved and loaded, there will be I/O from files too.

Classes, objects, methods:

- The main object is the GUI. It will have JPanel(s) inside it.
- I am NOT making the board its own object, nor am I making each piece its own object. I think that in this situation it would only add to the complexity of the program without make the program more elegant. *(For an example where each piece is an object, please see the BallWorld2 applet. If I were programming Battleship, I'd probably make each ship its own object.)*

FlowChart of Classes, Methods, Objects



File: Pente.java

- ♦ class Pente This class does all of the calculations.

  //main global variables:
  static board[][];
  static turn;

  - ❄ void main(String[] args) //args not used
    - makes a PenteGUI object

  - ❄ void processClick(int x, int y)
    - is the position already occupied?
    - update board array with new piece
    - determine if the game has been won
    - switch turn
    - call screen repaint
    - if AI, take AI turn
    - determine if the game has been won
    - repaint
    - switch turn
    - calls boardAction when game over ...

File: PenteGUI.java

- ♦ class PenteGUI This class does all of the graphics.

  - ❄ PenteGUI() //constructor
    - makes a PenteGUI object
    - makes a JFrame
    - adds menus
    - adds listeners for the menus
    - adds a DrawingPanel object
    - finish up by pack() and setVisible(true)

  - ❄ methods for menu listeners: quit, new game, etc.
    - ideally, this listener will just pass the menu name to the appropriate method in the Pente.java file.

- class DrawingPanel //inner class of PenteGUI
  - DrawingPanel() //constructor
    - creates object
    - adds listeners for mouse clicks
  - method for mouse listener
    - calculates location clicked
    - ignore if the click is ambiguous (e.g. too close to a line)
    - calls processClick() in Pente.java file
  - void paintComponent (Graphics g) //override JPanel paintComponent
    - the screen is automatically cleared by this method
    - paints screen
    - draws board
    - draws markers on board based on contents of board[][] array
  - void boardAction (int action)
    - does things like prevent clicking when the game is over, clearing the board & screen, ... (other?)

Other:

- boardAction() may not be needed for anything. It can be incorporated into the other DrawingPanel methods.
- no threads or timers are being used (since it's a turn-based game)
- no exceptions are thrown or handled,
  except for sleep(), which is trivial,when the AI player takes a turn.

Some ideas for a Java ISP

- battleship
- webspider
- a game that you play over two networked computers
- empire game
- scrabble
- some sort of animation (e.g. pong)
- pacman
- make an android app
- adventure game
- addressbook
- lego robot (lejos)
- paint program
- expand the Mandelbrot program to do more.

Possibly too easy:

- go
- tetris (on internet)
- scientific calculator
- 3D rotating cube (on internet: http://profs.etsmtl.ca/mmcguffin/learn/java/11-3d/)