Part I

1. Implementation

We implement beginner using Java. The idea goes like this:

```
class beginner
{
```

Members:

1) Parameters about the board information like # of rows, # of columns;
2) Parameters describing the state of the game, like playing, draw, player win, and agent win. The state of playing can be divided into player's turn and AI's turn;
3) Parameters to control the display.

Constructor of the class, which initializes the game and runs the whole game in a while loop. Later, the game will be initialized by instantiation the class.

Private methods:

1) `void initGame()`
   Initialize the game-board contents and the status
2) `void updateGame(Seed theSeed, int rowSelected, int colSelected)`
   Check the board after the player labeled as "theSeed" has placed on (rowSelected, colSelected), and update the game status.
3) `boolean isDraw()`
   check if the current game status is a draw.
4) `boolean hasWon(Seed theSeed, int rowSelected, int colSelected)`
   Return true if the player with "theSeed" has won after placing at (rowSelected, colSelected)
5) `boolean open3InARow(Seed theSeed)`
   check if there exists the case open 3-in-a-row for the player or the AI. If yes, place on the 4th position to block or to win.
6) `void PlayerMove()`
   Move strategy for the player. Player cannot place on the position where is not empty. It is activated by a mouse event.
7) `void AIMove()`
8) Move strategy for the AI. AI cannot place on the position where is not empty. It will check if there exists a open-3-in-a-row situation. If yes, response accordingly. Otherwise, randomly place on the board.
9) `void visualSet()`
   Display settings like the status bar, title, and showing the board.

Inner class:

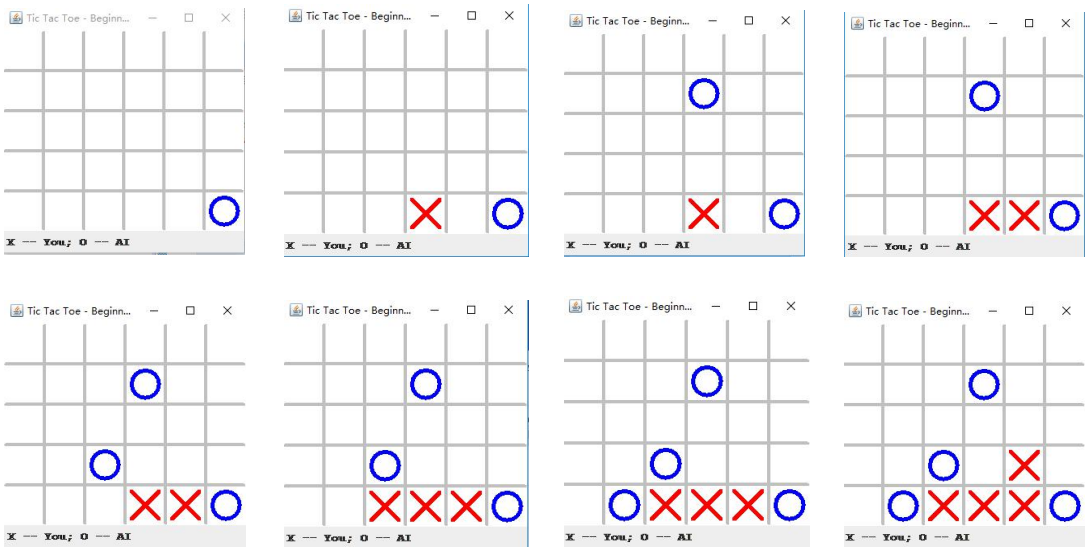
`class DrawCanvas extends JPanel` is responsible to draw the board, and the

markings for the two players.

2. Output

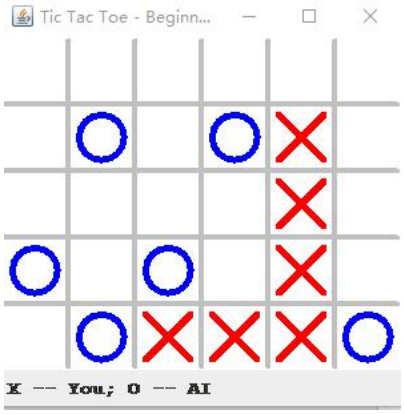
Since some moves of the agent is random, the output of the game is unrepeatable. Following is first 8 moves in one game we have played.

First 8 moves:



'O' in the board denotes the markings of the agent, and 'X' on the board denotes the markings of the player. In the 7th step, the agent chooses to block the player rather than randomly place its markings, which is also our expectation.

Final board and win of the game are shown below:



Part II & II & IV

1. Implementation

In this part, We implement Beginner,Advanced,Master,Board in java. The basic ideal goes like following. Even though we don't have a class name master, we can change the value of depth in this function miniMax in class Advanced to create a master.

Class Advanced {
Public int[] miniMax(Board state, int depth) {
When depth is 2, it represents advanced, while if depth is 4, it represents 4
}
Public int[][] possibleActions(Board state) {
This function return all the actions current state can take, given a current state
}
Public Board result(Board state), int[] actions, String pattern) {
This function return a new chess board, when given a current state, an action, and player(pattern means player, just like "O","Y","X")
}
Public int minValue(Board state, int depth) {
This function return the min value of current state, given the depth.
}

Public int maxValue(Board state, int depth) {
This function return the max value of current state, given the depth
}
}

Class Board{
String[][] configuration = new String[5][6]; this represent the chess board situation

```
    int openThreeAdvanced = 0;
    int openTwoAdvanced = 0;
    int openThreeBeginner = 0;
    int openTwoBeginner = 0;
    int openThreeMaster = 0;
    int openTwoMaster = 0;
    int depth = 0;
```

Public void setParamter() {
This function set get class variables above, when given the chess board configuration(configuration means which block is occupied by which player)
}
Public String getDiagonal1(int[] position) {
This function return a diagonal(string) when given a start point, from leftUp to

rightDown
}
Public String getDiagonals(int[] position)
This function return a diagonal(string) when given a start point, from rightUp to leftDown
}

Public boolean isOpenThree(String str, string pattern) {
This function return whether a string has a open-3-in-line of a pattern('O','Y','X')
}

Public int isOpenThree(String str, string pattern) {
This function return the number of open-2-in-line of pattern in a string
}
Public boolean hasBlank() {
This function return whether there a space left in the chess board
}

Public int gameOver() {
This function return different number which can indicates whether the game is over or which player wins the game.
}

Class Beginner() {
Public Board randomMarking(Board board) {
This function return a new Board formed by taking a randomly action
}

Public Board markingToGetFourInRow(Board board) {
This function return a new Board formed by taking an action to make one player win
}

Public Board markingToPresentFourInRow(Board board) {
This function return return a new Board formed by taking an action to present one player win
}
}

Class StartHere {
```
public int beginnerVSadvanced() {
```

```
    I call this function in main to achieve the function of Part II in homework.
    As its' name shows, it function is to hold a game between beginner and
    advanced,beginner move first.
}
public int beginnerVSmaster() {
As its' name shows, it function is to hold a game between beginner and
master,beginner move first.
}

public int advancedVSmaster() {
 I call this function in main to achieve the function of Part III in homework.
 As its' name shows, it function is to hold a game between advanced and
  master,advanced move first.
  }

public int advancedVSbeginner() {
   Same as beginnerVSadvanced(), but advanced move first
}

public int masterVSbeginner() {
    Same as beginnerVSmaster, but master moves first
}
public int masterVSadvanced() {
  Same as advancedVSmaster, but master moves first
}
public void tournament() {
  I call this function in main to achieve the function of part IV in homework.As
  its' name shows, it holds 100 games for each pair of players.
}
}
```

To implement Part II, I create an instance of class Beginner called beginner and an instance of class Advanced called advanced, I call the function of these two objects in a while loop until one object wins or there is a tie. According to the current situation, I call the function of beginner, like randomMarking, markingToGetFourInRow, markingToPresentFourInRow respectively. As for advanced, I call the function miniMax(state,2), 2 means 2 step advanced should looking forward.

To implement Part III, it is almost the same as the implementation of Part II. The only difference is that I call the function miniMax(state,4) very time when master move(4 means looking forward 4 steps).

To implement Part IV, I create a beginner , a advance, a master. I call 100 time for each pair of player and use arrays to store the result of the competition.

2. Output

Part II

First 8 moves .O represents beginner. X represents advanced. M represents blank

```
MOMMMM
MMMMMM
MMMMMM
MMMMMM
MMMMMM
```

```
XOMMMM
MMMMMM
MMMMMM
MMMMMM
MMMMMM
numebr of nodes : 841
it costs : 47 ms
```

```
XOMMMM
MMMMMM
MMMMMM
MOMMMM
MMMMMM
```

```
XOXMMM
MMMMMM
MMMMMM
MOMMMM
MMMMMM
numebr of nodes : 729
it costs : 47 ms
```

```
XOXMMM
MMMMMM
MMOMMM
MOMMMM
MMMMMM
```

```
XOXXMM
MMMMMM
MMOMMM
MOMMMM
MMMMMM
numebr of nodes : 625
it costs : 31 ms
```

```
XOXXMM
MMMMMM
MMOMMM
MOMMMM
MMMMOM
```

```
XOXXXM
MMMMMM
MMOMMM
MOMMMM
MMMMOM
numebr of nodes : 529
it costs : 31 ms
```

Final board

```
XOXXXO
OXXXXM
MMOMMM
MOMMMO
MMOMOM
```

Advanced wins

Part III

First 8 moves, X represents advanced, Y represents master M represents blank

```
X M M M M M
M M M M M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 900
it costs: 62ms
```

```
X M M M M M
M Y M M M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 592789
it costs: 29729ms
```

```
X X M M M M
M Y M M M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 784
it costs: 38ms
```

```
X X Y M M M
M Y M M M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 439479
it costs: 22186ms
```

```
X X Y M M M
M Y X M M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 676
it costs: 25ms
```

```
X X Y M M M
M Y X Y M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 318025
it costs: 16109ms
```

```
X X Y X M M
M Y X Y M M
M M M M M M
M M M M M M
M M M M M M
number of nodes: 576
it costs: 17ms
```

```
X X Y X M M
M Y X Y M M
M Y M M M M
M M M M M M
M M M M M M
number of nodes: 222835
it costs: 11297ms
```

Final board

```
X X Y X X M
M Y X Y X M
Y Y Y Y X M
M M M M M M
M M M M M M
number of nodes: 49780
it costs: 2344ms
master wins
```

Part IV

Play first

	Beginner first	Advanced first	Master first
Beginner		Advanced Win 46 Advanced Loose 3 Tie 1	Master Win 50 master Loose 0 Tie 0
Advanced	Beginner Win 7 Beginner Loose 34 Tie 9		Master win 50 Master loose 0 Tie 0
Master	Beginner Win 0 Beginner Loose 49 Tie 9	Advanced win 0 Advanced loose 50 Tie 0	