



**Othello puzzle game using  
Netbeans, Java Swing and Artificial  
Intelligence**

# Who am I

- Graduate of Athens University of Economics and Business, Computer Science
- Software Developer
- Just finished my military service
- Thesis: Android Application, Augmented Reality for the Museum of Telecommunications (OTE – COSMOTE)
- M.Sc in Computer Science Student

# Project Parts

- Game Play
- Artificial Intelligence
- Graphical Interface with Java Swing

## **What is Othello / Reversi ?**

Reversi is a strategy board game for two players, played on an 8×8 uncheckered board.

Players take turns placing disks on the board with their assigned color facing up.

During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

# Gameplay

Three (3) basic Classes

- Board.java
- Move.java
- GamePlayer.java

# Board

Game Board is actually an Integer 8x8 - Array .

We set the moves and the Rules of the game.

Every board game has bounds so we check if a player's move is *In Bounds*.

There is also a method that checks if the move is *Valid* (for example check if the position is empty and the move is valid according the game rules)

# Artificial Intelligence

- **Min-Max algorithm**

Min-Max is a decision rule used in decision theory and game theory. This algorithm is used for minimizing the possible loss for a worst case scenario. Min-Max is searching a tree and tries to choose the best move (the best Node). What is the best move?

## **Evaluate Method**

With this Method we evaluate every move. Every move has a score. For example the Corners of Othello's board are very important and we evaluate them as a very (high score) nice move.

At every turn of the Game we evaluate the moves again and Min-Max searches all the possible plays and tries to choose the best Move.

# Hard Level

Min-Max algorithm calculates the best move. If you set the algorithm to search in *big depth* (search the Min-Max tree in depth - HARD LEVEL) the results will be better than searching in smaller depth. The computer will choose better moves that will be difficult to be beaten. If you set the algorithm to search in *small depth* the results will not be the best (EASY LEVEL).

HARD LEVEL isn't difficult only for the human player, but also for the computer player. The computer has to do more calculations to find the best move and the game's efficiency is reduced. Sometimes when it is the turn of the computer, there are big delays (especially at the start of the game).

Solution ▯ Alpha – Beta Pruning



# Alpha – Beta Pruning

***Alpha – Beta pruning*** is a search algorithm that seeks in decreasing the number of nodes that are evaluated by the *Min – Max algorithm* in its search tree. It stops evaluating a move when at least one possibility has been found that the move is worse than a previously examined move. Such moves need not to be further evaluated.

# Java Swing

`Gui.java` class contains most of the graphic methods of the game. Java swing's 2D graphics have also been used.

```
import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
```

```
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
```

# Graphical Interface

Each empty space of board is actually a `Button Icon` (`BufferedImage` object). An icon (an empty green image-field) with the characteristics of a button. When the player chooses a `VALID` position on board the game sets players disk (sets a disk icon over the “empty” green field) and changes all disks according to the games rules.

# Important Methods and Functions

`public Gui(int level, int color)`

`public static void initializeOthello(int level, int color)`

`public static boolean playerMove(ActionEvent a)`

`public static boolean computerMove()`

`public static void refreshTable()`

`public static boolean endgame()`

# Important Java Swing Objects and the Methods

```
GridLayout grid = new GridLayout(8,8);
```

```
ImageIO.read
```

```
.setIcon(new ImageIcon(image));
```

```
.addActionListener(this); (every position of the grid has a  
Listener and reacts on mouse click)
```

# Menu and Menu Bar

Java Swing provide us libraries for menu and menu bars. We created a simple menu (primary menu) with the following options:

- Select color
- Select level of difficulty

This menu is a `JFrame` and contains `JRadioButtons` (Selects) and a Start ("Play") button.

# Important Objects and Methods of Primary Menu

JFrame():

- `getContentPane().setFont`
- `.setBounds(50, 50, 700, 700);`
- `.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`

JRadioButton:

- `.setFont(new Font("Dialog", Font.PLAIN, 14));`
- `.setActionCommand("Easy");`
- `.getSelection().getActionCommand().equals("Easy")`

# The Othello®



Select colour:

☒ Black

☐ White

Select level of difficulty:

☐ Easy

☐ Medium

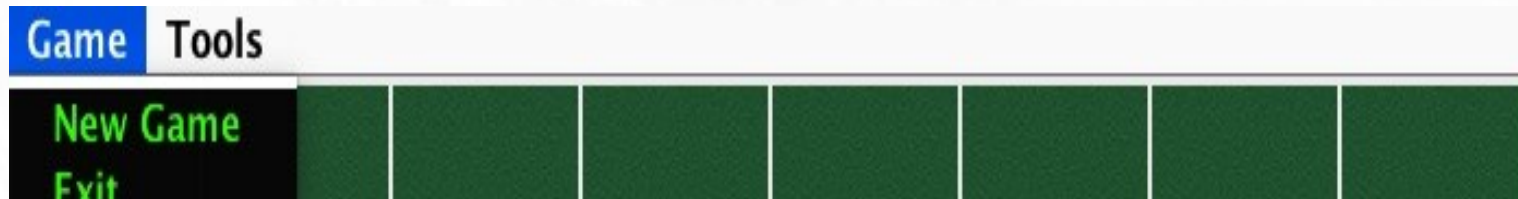
☐ Hard



**PLAY**



# Menu Bar



A simple menu bar which contains the options:

- New Game
- Score
- Exit

# Important Objects and Methods of Menu Bar

JMenuBar();

JMenu("Game");

JMenuItem("New Game");

# Score Window

`Score.java` is an extra `JFrame` that displays the score of the game. It's a simple window where we count and show the game score the specific moment.

A method (in `Board.java`) named `public int ScoreCount(int player)` calculates the score of each player.

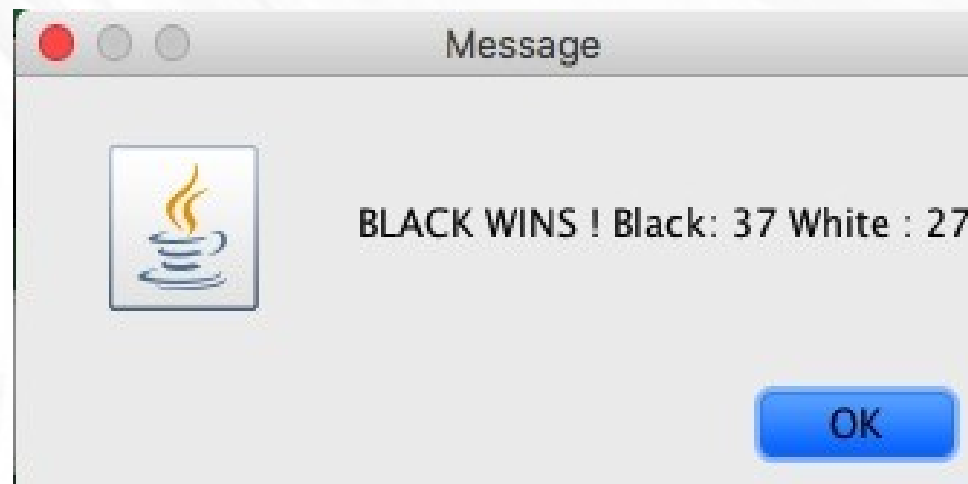


# showMessageDialog

```
JOptionPane.showMessageDialog(diag, "Your message here");
```

it's an easy way to create a simple graphical window with a message for the user. It is just one line of code with Java Swing using this library:

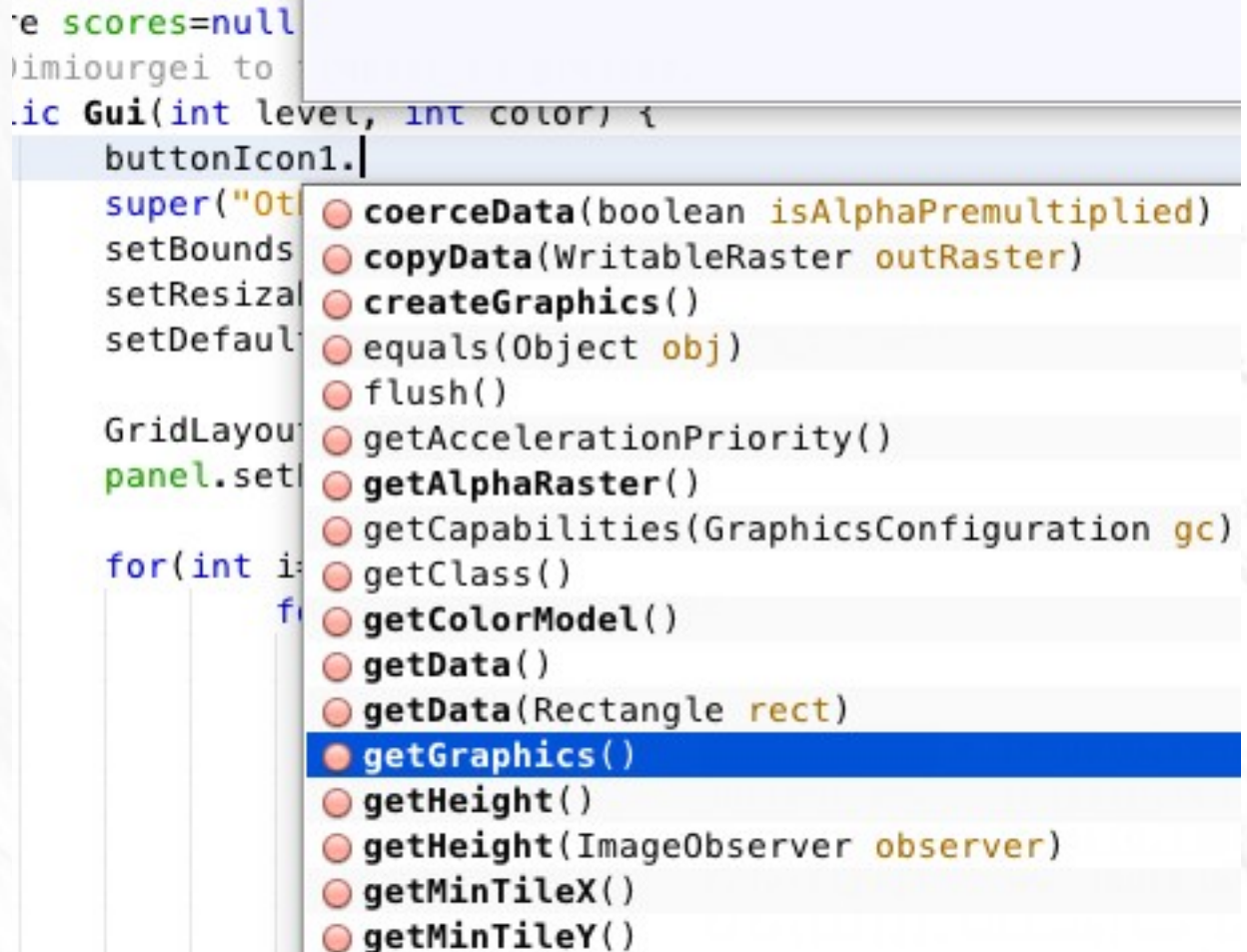
```
import javax.swing.JOptionPane;
```



# NetBeans : A helpful Assistant (1)

Netbeans is a useful assistant for a programmer.

Netbeans helps us to organize our source code especially when our code is big and we have many classes to manage.



The screenshot shows a Java code editor with a list of methods suggested by the NetBeans IDE. The code in the background includes:

```
...e scores=null
...imiourgei to
...ic Gui(int level, int color) {
    buttonIcon1.
    super("Ot
    setBounds
    setResizal
    setDefault
    GridLayout
    panel.set
    for(int i
    f
```

The completion list on the right contains the following methods:

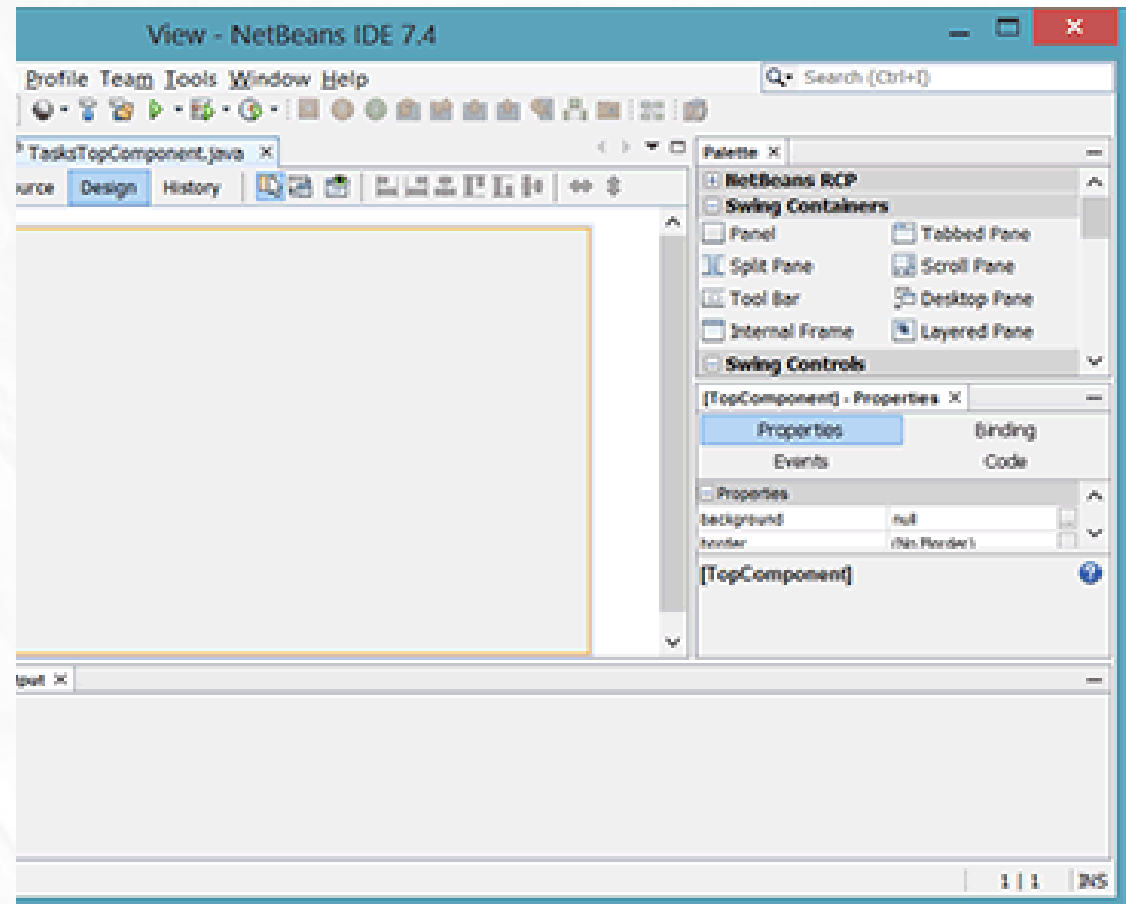
- coerceData(boolean isAlphaPremultiplied)
- copyData(WritableRaster outRaster)
- createGraphics()
- equals(Object obj)
- flush()
- getAccelerationPriority()
- getAlphaRaster()
- getCapabilities(GraphicsConfiguration gc)
- getClass()
- getColorModel()
- getData()
- getData(Rectangle rect)
- getGraphics()** (highlighted)
- getHeight()
- getHeight(ImageObserver observer)
- getMinTileX()
- getMinTileY()

# NetBeans : A helpful Assistant (2)

Netbeans is very useful when we have to implement graphical interface. Using Matisse (the NetBeans GUI Builder) but also without it.

Netbeans editor contains ALL Java Swings methods, explains each method and imports every library automatically.

Code generation (Insert Methods like getters and setters)



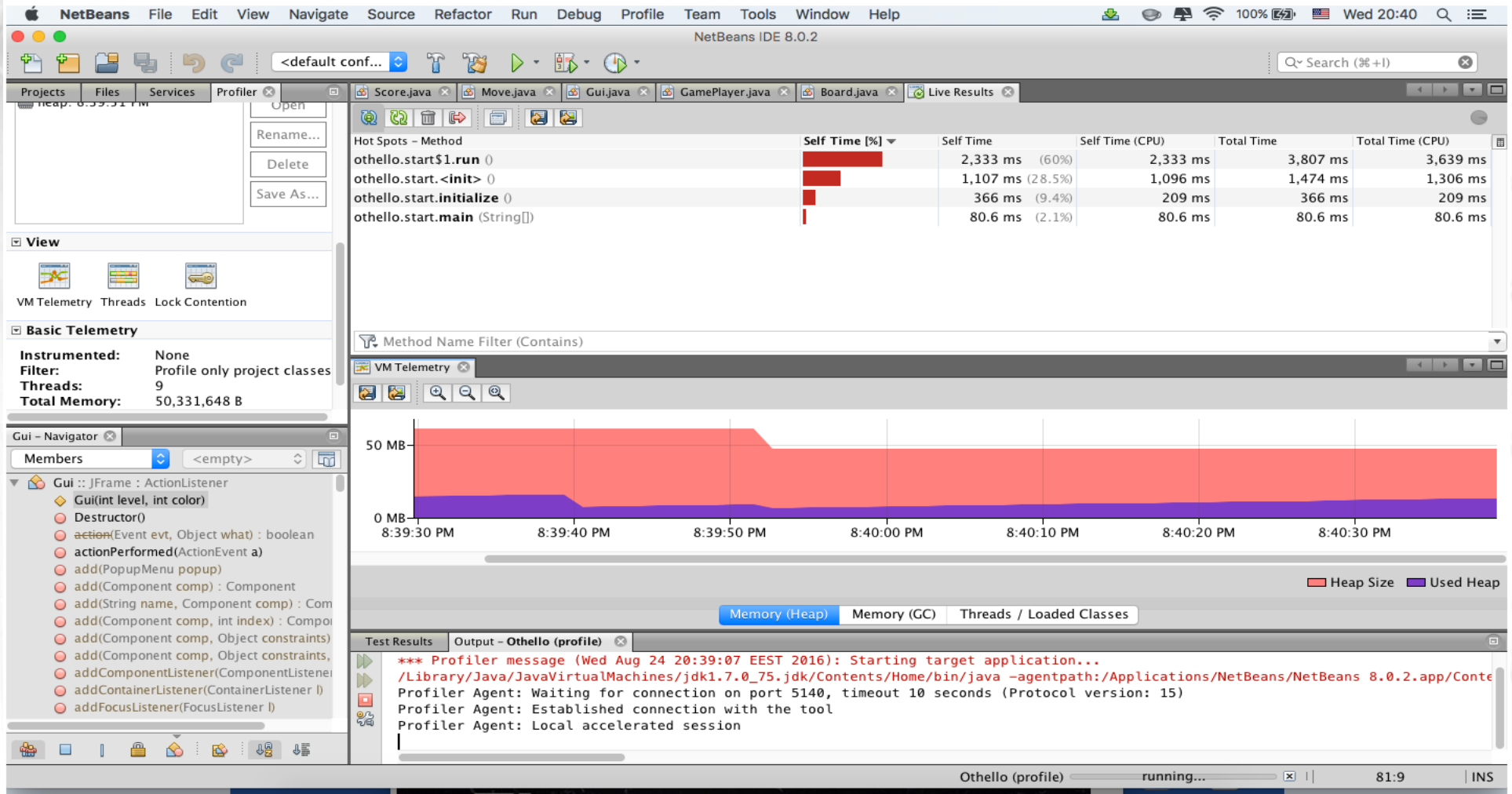


# NetBeans : A helpful Assistant (3)

With Netbeans we can easily find errors, build and run our code. Using Profiler we can check your cpu time delay when we run our app and also check memory usage.

We can also build an executable .jar file and just double click the file to run the game.

# NetBeans : A helpful Assistant (4)





# Questions and Answers



**Thank you!**