**1.0 Introduction**

Java is a [general-purpose](http://en.wikipedia.org/wiki/General_purpose_programming_language), [class-based](http://en.wikipedia.org/wiki/Class-based), [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) [computer programming language](http://en.wikipedia.org/wiki/Computer_programming_language) applications are typically [compiled](http://en.wikipedia.org/wiki/Compiler) to ([class file](http://en.wikipedia.org/wiki/Class_%28file_format%29)) that can run on any [Java virtual machine](http://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of [computer architecture](http://en.wikipedia.org/wiki/Computer_architecture). As 2013, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users. Java was originally developed by [James Gosling](http://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) which has since [merged into Oracle Corporation](http://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle) and released in 1995 as a core component of Sun Microsystems' [Java platform](http://en.wikipedia.org/wiki/Java_%28software_platform%29). The language derives much of its [syntax](http://en.wikipedia.org/wiki/Syntax_%28programming_languages%29) from [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [C++](http://en.wikipedia.org/wiki/C%2B%2B), but it has fewer [low-level](http://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them.

The task assigned is to design a MasterMind board game in java programming. This task will be performed by utilizing the java-programming platform. More details will be discussed in further sections.

**2.0 Objective of the assignment:**

* Design an attractive, effective and user-friendly program.
* Interactivity /Randomness / unpredictability for game-play.
* Exception-handling.
* Final program meeting all the requirements.

**3.0 Project overview:**

Originally Mastermind is a board game, and it’s generally in the form of a perforated 10 rows of four holes. Those are the holes to accommodate the different color pins.

The number of pieces of different colors is 8 and are usually eight colors: red; yellow; green; blue; orange; off white; purple; fuchsia. Number of colors can vary depending on the difficulty.

There are also red and white (or black) pins used to provide guidance at each stage of the game. There are many variations on the number of colors or rows of holes.

A player begins by placing his choice of pieces without being seen from the other player in the back of a cache that hide at the sight of it until the end of the round, or in our case the program won’t reveal the combination.

If the player has not selected the pieces must find out what are the four pieces, that is to say their colors and positions by clicking on the holes. At every turn, the player must choose the combination to fill a row according to his idea of the hidden pieces. Once the pieces placed, the other player says:

1. The number of pieces of good well placed pieces.
2. The number of pieces of the right color but the wrong place, with the white pieces.

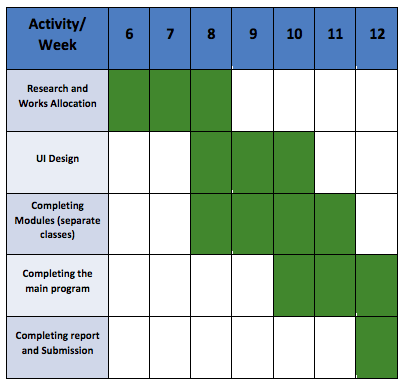
A declaration has to be said after the player places his pins that no pawn match, in color or color and position if that’s true. The next placement by the player has to take in account the previous results.

The tactics of the active player is to select based on previous strokes, colors and positions in order to obtain the maximum information from the response of the partner as the number of applications is limited by the number of rows of holes in the game. In most cases, he tries to get as close as possible to the solution, given the previous answers, but it can also form a combination for the sole purpose of verifying some of the conclusions of previous moves and to accordingly the proposal is more conducive to the deduction of new information.

The player wins this round if he gives the right combination of pieces on the back row or front. In any case, it is his turn to choose the pieces to discover.

**3.1 Project plan:**

Project Gannt chart



Task Allocation

|  |  |
| --- | --- |
| Name | Tasks |
| Raf | * Integrate basic Mastermind game with GUI (GamePanel) |
| Aziz | * Writing basic Mastermind game module (text based) * Write timer and score modules * Write sound and music modules |
| Aliffah | * Design GUI (overall) * Write highscore module * Design and write HowtoPlayPanel class * Design and write HighscorePanel class |

**4.0 Project Design:**

The game panels and design are as the following sketches:

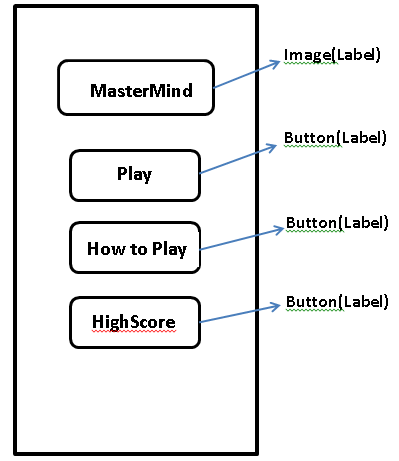


Figure 1: Main Menu

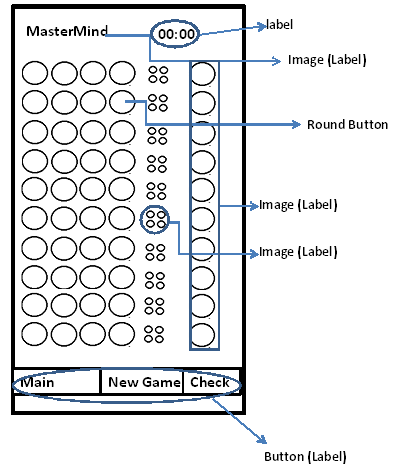


Figure 2: Game Panel

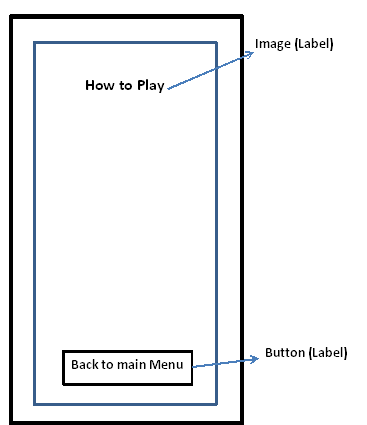


Figure 3: How to Play Panel

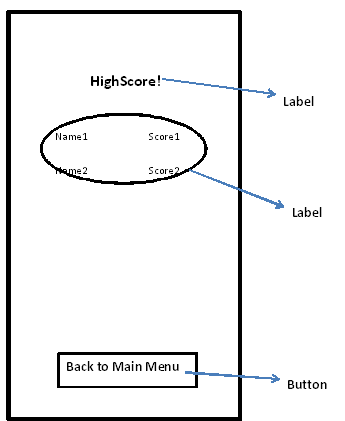


Figure 4: High Score

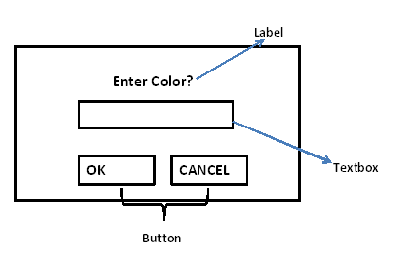


Figure 5: Dialog Box

**5.0 Task Assigned:**

**Playing Sound:**

Playing a sound or music of wav format. The class allows the user to play a sound from a wav file. Sound playback is encapsulated in a thread, which allows immediate access for many situations. For example, in the development of a JAVA MasterMind game, the sound is played in the background and keeps looping.

The source code is a basic class called **LetsPlayWav**. Which is able to play a wav format file, some other formats are also supported. The sound will be played asynchronously in a thread without interruption of the main program. This feature will very useful in the development of the game as we don’t want the program to hang while playing the background music, it also can be modified to be used for sounds at events such as a click of a button.

This class extends a java thread to be asynchronous. This calss has two constructors LetsPlayWav(string wavfile) and LetsPlayWav(string wavefile, position p) the position can be LEFT of RIGHT for the stereo sound system.

**javax.sound.sampled** classes are used to prepare for preparing the audio data input stream to be stored in a buffer and from there to be processed.

import *java.io.File*;

import *java.io.IOException*;

import *javax.sound.sampled.AudioFormat*;

import *javax.sound.sampled.AudioInputStream*;

import *javax.sound.sampled.AudioSystem*;

import *javax.sound.sampled.DataLine*;

import *javax.sound.sampled.FloatControl*;

import *javax.sound.sampled.LineUnavailableException*;

import *javax.sound.sampled.SourceDataLine*;

import *javax.sound.sampled.UnsupportedAudioFileException*;

***// playing sound in the background from a wave file***

public class LetsPlayWave extends Thread {

private String filename;

private Position curPosition;

private final **int** EXTERNAL\_BUFFER\_SIZE = 524288; *// 128Kb*

enum Position {LEFT, RIGHT, NORMAL};

public LetsPlayWave(String wavfile) {

filename = wavfile;

curPosition = Position.NORMAL;

}

public **void** run() {

File soundFile = new File(filename);

if (!soundFile.exists()) {

System.err.println("Wave file not found: " + filename);

return;

}

AudioInputStream audioInputStream = null;

try {

audioInputStream = AudioSystem.getAudioInputStream(soundFile);

} catch (UnsupportedAudioFileException e1) {

e1.printStackTrace();

return;

} catch (IOException e1) {

e1.printStackTrace();

return;

}

AudioFormat format = audioInputStream.getFormat();

SourceDataLine auline = null;

DataLine.Info info = new DataLine.Info(SourceDataLine.class, format);

try {

auline = (SourceDataLine) AudioSystem.getLine(info);

auline.open(format);

} catch (LineUnavailableException e) {

e.printStackTrace();

return;

} catch (Exception e) {

e.printStackTrace();

return;

}

if (auline.isControlSupported(FloatControl.Type.PAN)) {

FloatControl pan = (FloatControl) auline.getControl(FloatControl.Type.PAN);

if (curPosition == Position.RIGHT) {

pan.setValue(1.0f);

} else if (curPosition == Position.LEFT) {

pan.setValue(-1.0f);

}

}

auline.start();

**int** nBytesRead = 0;

**byte**[] abData = new **byte**[EXTERNAL\_BUFFER\_SIZE];

try {

while (nBytesRead != -1) {

nBytesRead = audioInputStream.read(abData, 0, abData.length);

if (nBytesRead >= 0) {

auline.write(abData, 0, nBytesRead);

}

}

} catch (IOException e) {

e.printStackTrace();

return;

} finally {

auline.drain();

auline.close();

}

}

}

auline.start();

**int** nBytesRead = 0;

**byte**[] abData = new **byte**[EXTERNAL\_BUFFER\_SIZE];

try {

while (nBytesRead != -1) {

nBytesRead = audioInputStream.read(abData, 0, abData.length);

if (nBytesRead >= 0) {

auline.write(abData, 0, nBytesRead);

}

}

} catch (IOException e) {

e.printStackTrace();

return;

} finally {

auline.drain();

auline.close();

}

}

}

**Timer Class:**

The timer class is a class that can be initialized and count the time. It doses that by keeping the count at every time step of x. the below code is the code tested for this class which has a GUI and an action listener. From there we can integrate the timer class with our main game program.

import javax.swing.\*;

import javax.swing.border.\*;

import java.awt.\*;

import java.awt.event.\*;

public class DTimer extends JFrame implements ActionListener

{

//Attributes:

protected JLabel viewTime; // Component for displaying the time elapsed

protected int timeCount; // Variable to memorize the elapsed time

protected Timer timer; // objet javax.swing.Timer

//Constructor :

// (Default) sets the counter to 0 and a 1 second delay

public DTimer ()

{ this (0, 1000);

}

// Built a timer with the given initial time and the delay to 1 second

public DTimer (int initialTime)

{ this (initialTime, 1000);

}

// Built a timer with time and the delay gives

public DTimer (int initialTime, int delay)

{ super ("Demo clock without thread");

this.timeCount = initialTime;

this.viewTime = new JLabel (""+this.timeCount);

this.timer = new Timer (delay, this);

this.initGraphic ();

}

//Methods :

// Initialize the graphical part of the demo

protected void initGraphic ()

{ JButton bouton = new JButton ("start/stop");

bouton.setActionCommand ("Bouton");

bouton.addActionListener (this);

this.viewTime.setHorizontalAlignment (JLabel.CENTER);

this.getContentPane ().add (bouton, BorderLayout.SOUTH);

this.getContentPane ().add (this.viewTime, BorderLayout.CENTER);

this.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);

this.setLocation (250, 250);

this.pack ();

this.setVisible (true);

}

// Starts the timer

public void startDTimer ()

{ this.timer.start ();

}

// Stop the timer

public void stopDTimer ()

{ this.timer.stop ();

}

// Used to recover the time already elapsed

public int getTime ()

{ return ( this.timeCount );

}

// Allows to know the state of activity timer (spear or not)

public boolean isRunning ()

{ return ( this.timer.isRunning () );

}

public void actionPerformed (ActionEvent e)

{

// Case generates an event with the button

if ( "Bouton".equals (e.getActionCommand ()) )

{

if ( this.isRunning () )

{ this.stopDTimer ();

}

else

{ this.startDTimer ();

}

}

else // Case generates an event by the component javax.swing.Timer

{ this.timeCount++;

this.viewTime.setText (""+this.timeCount);

}

}

public static void main (String argv [])

{ new DTimer ();

}

}

// Stop the timer

public void stopDTimer ()

{ this.timer.stop ();

}

// Used to recover the time already elapsed

public int getTime ()

{ return ( this.timeCount );

}

// Allows to know the state of activity timer (spear or not)

public boolean isRunning ()

{ return ( this.timer.isRunning () );

}

public void actionPerformed (ActionEvent e)

{

// Case generates an event with the button

if ( "Bouton".equals (e.getActionCommand ()) )

{

if ( this.isRunning () )

{ this.stopDTimer ();

}

else

{ this.startDTimer ();

}

}

else // Case generates an event by the component javax.swing.Timer

{ this.timeCount++;

this.viewTime.setText (""+this.timeCount);

}

}

public static void main (String argv [])

{ new DTimer ();

}

}

the time counted using the timer class is also a factor for the score algorithm.

Below is the sample code that do this feature.

algorithm for score

newscore = ((11 - guessCount)\*100) - timeCount\*10; //score algorithm

**6.0 Conclusion:**

By doing this assignment, we were able to design a Java program to provide a user-friendly interface, which allows the user to play MasterMind game. Also adding some features to it such as a moderate difficulty level, guidance in how to play the game and timer function to count the time taking to play the game. Furthermore, experience in Java programming platform and familiarity with the java syntax has been gained. In addition to the programming experience and the practice in java programming we have gained a lot from just by the team work it self. At the end of this assignment all of the objectives has been achieved and successfully completed.