## THE DESIGN AND IMPLEMENTATION OF A CHESS GAME USING JAVA

# BACHELOR OF TECHNOLOGY

### IN

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**N DHANA LAKSHMI (18C91A0560)**

**M PRAMOD (18C91A0553)**

**K PRAVIN (18C91A0543)**

**Under the Esteemed guidance of**

Mrs.CH.TULASI RATNA MANIM.Tech

Assistant professor



### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE**

**(COLLEGE OF ENGINEERING)**

***(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with ‘A’ Grade)*Bogaram (V), Keesara (M), Medchal District -501 301.**

### 2021 - 2022

**HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE**

#### (COLLEGE OF ENGINEERING)

***(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with ‘A’ Grade)***

**Bogaram (V), Keesara (M), Medchal Dist-501301.**

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



**CERTIFICATE**

## This is to certify that the mini project entitled “THE DESIGN AND IMPLEMENTATION OF A CHESS GAME USING JAVA”isbeingsubmittedby**N DHANA LAKSHMI (18C91A0560)**,**M PRAMOD (18C91A0553), K PRAVIN(18C91A0543),**inPartial fulfillment of the academic requirements for the award of the degree of Bachelor ofTechnologyin“COMPUTERSCIENCEANDENGINEERING”HOLYMARYINSTITUTEOFTECHNOLOGY & SCIENCE, JNTU Hyderabad during the year 2021- 2022.

**INTERNALGUIDE HEAD OF THEDEPARTMENT**

Mrs.CH.TULASI RATNA MANI M.TechDR .B.NARSIMHA M.Tech,Ph.D.

Assistant professorProfessor &HoD

Dept. of Computer Science&Engineering. Dept. of Computer Science &Engineering

### EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be in complete without the mention of the people whom adeit possible,who’sconstant guidance and encouragement crowns all effort withsuccess.

I take this opportunity to express my profound gratitude and deep regards to My Guide **Mrs.CH.TULASI RATNA MANI**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science for his / her exemplary guidance, monitoring and constant encouragement throughout the project work.

My special thanks to **Dr.B.Narsimha,HeadoftheDepartment**,Dept.of Computer Science & Engineering, Holy Mary Institute of Technology & Science who has given an immense support through out the course of theproject.

I also thank to **Dr. P. Bhaskara Reddy,** the **honorable Director** of my college Holy Mary Institute of Technology & Science for providing me the opportunity to carry out this work.

At the outset, I express my deep sense of gratitude to the beloved **Chairman A. Siddartha Reddy** of **Holy Mary Institute of Technology & Science**, for giving me the opportunity to complete my course of work

I am obliged to **staff members** of Holy Mary Institute of Technology & Science for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Last but not the least I thank **ALMIGHTY** and My **Parents**, and **Friends** for their constant encouragement without which this assignment would not be possible.

**N DHANA LAKSHMI (18C91A0560)**

**M PRAMOD (18C91A0553)**

**K PRAVIN (18C91A0543)**

**DECLARATION**

## This is to certify that the work reported in the present project titled **“**THE DESIGN AND IMPLEMENTATION OF A CHESS GAME USING JAVA**”** is a record of work done by me in the Department of Computer Science & Engineering, Holy Mary Institute of Technology and Science.

No part of the thesis is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text the reported are based on the project work done entirely by me not copied from any other source.

**N DHANA LAKSHMI (18C91A0560)**

**M PRAMOD (18C91A0553)**

**K PRAVIN (18C91A0543)**

CONTENTS

ABSTRACT

Chapter Name of the Chapter PageNo.

### INTRODUCTION

* 1. ProblemStatement 8
  2. Objectives 8
  3. Motivation 8
  4. ExistingSystem 9
  5. ProposedSystem 9

### LITERATURESURVEY

* 1. ExistingSystem 10
  2. ProposalSystem 10
  3. Summary 11

1. **SOFTWARE REQUIREMENTS SPECIFICATIONS**
   1. SoftwareRequirements 13
   2. HardwareRequirements 13

### SYSTEMDESIGN

* 1. SystemArchitecture 14

### IMPLEMENTATION

* 1. EnvironmentalSetup 15
  2. ModuleDescription 25
  3. SoftwareDescription 25
  4. SampleCode 25

### SYSTEMTESTING

* 1. Tests 42

### RESULTSCREENSHOTS 43

### CONCLUSION 48

### BIBLIOGRAPHY 49

LIST OFSCREENSHOTS

1. Window for setting the player names and time 43
2. The chess board after starting game 44
3. Instructions 45
4. New game 46
5. Game over messages 47
6. **INTRODUCTION**

**ProblemStatement:**THE DESIGN AND IMPLEMENTATION OF A CHESS GAME USING JAVA.

**Objectives:**

* Our objective was to develop and test a bug-free standalone game in java, complete with a GUI and game logic components.
* Nowadays, many children are addicted to the computer games, and tends to neglect the traditional games like chess. The chess game is designed in order to attract the interest of the children towards the chess game.
* We developed a bug-free, fast and well-designed product with a clean user interface.
* This game is built using core java, java swing GUI libraries and the Unit test suite. It uses custom drawing for game components and self-programmed logic for checkmate detection.

**Motivation:**

Chess can be a very motivating game - at least, I find, when you win! But it's very painful when you find yourself being ground down and inevitably in a position where you are going to lose. You know it - you can see it from far off - you try to stop it - you try to reverse that bad position you are in, but your opponent relentlessly takes advantage of your weakened condition, exploits the gaps in your defence, and ultimately out-thinks you to defeat. So, you sit looking, sadly, at what might have been. Ah!

So chess is motivating because ultimately it is about personal responsibility - about accepting that responsibility and not making excuses for your game. In terms of the nine motivators that account for what people want it is predominantly a Spirit motivator - chess expresses primarily our need to be free to take the board constraints and dominate the 'space' we create.

Of course, once that freedom is achieved other motivators also come into play: the victory - oh yes! - the Star motivator - recognition big time. And the Expert motivator : the geeks study for this. And let's not forget the Friend motivator - the belonging to the clubs and sharing one's passions and enthusiasm with others. Yes, it's all there in chess.

Chess, then, is not only a game of intellect, it is also a game of psychology, and without it I think it must be impossible to win.

But to return to the theme of motivation: whilst losing isn't great, I have to say that chess is motivating, win or lose. There is something about the game that is intrinsically gripping and beneficial. This is essentially to do with the fact that chess is one of those few games where luck plays absolutely no part: many card games involve high levels of skill, but there is always the luck of the deal, and sometimes even the most skilful players can have an awful run of cards. But with chess every move is your own and has to be considered within the context of the very definite rules.

**ExistingSystem:**

* Here they need to play the game manually. Where two people play the chess game on the normal chess board.
* The online chess game apps provide timer for each move or timer together but they does not provide Timer for each player for the whole game so some people waste their time for manipulating the other player where as in original competitions the each player is provided with limited amount of time for the total game not for each move.

**ProposedSystem:**

* The main objective of Chess game is to enhance and upgrade the existing System increasing its efficiency and effectiveness. The software improves the working methods replasing the existing manual system with computer based system.
* Proposed system will give a new and bright future to the Chess game.
* Developing the chess game.
* By adding Timer to it in such a way that time is allocated individually to each player and time allocated is equal for both the players.
* If a player waste his time(when one player’s time reaches Zero) then automatically the opponent will be declared as the winner.
* The followings will be the procedure for the system…The PC programming will naturally plot the graphical portrayal of the genuine chess board, for all intents and purposes. This information will be sent to the PC to PC using protocols through cable.

2.LITERATURESURVEY

**ExistingSystem:**

* Here they need to play the game manually. Where two people play the chess game on the normal chess board.
* The online chess game apps provide timer for each move or timer together but they does not provide Timer for each player for the whole game so some people waste their time for manipulating the other player where as in original competitions the each player is provided with limited amount of time for the total game not for each move.
* So Time wastage is the major issue in Existing System.

**ProposalSystem:**

* The main objective of Chess game is to enhance and upgrade the existing System increasing its efficiency and effectiveness. The software improves the working methods replacing the existing manual system with computer based system.
* Proposed system will give a new and bright future to the Chess game.
* Developing the chess game.
* By adding Timer to it in such a way that time is allocated individually to each player and time allocated is equal for both the players.
* If a player waste his time (when one player’s time reaches Zero) then automatically the opponent will be declared as the winner.
* The followings will be the procedure for the system…The PC programming will naturally plot the graphical portrayal of the genuine chess board, for all intents and purposes. This information will be sent to the PC to PC using protocols through cable.

**Summary:**

* The methodology of chess implementation has been described in this section.
* The main objective of Chess game is to enhance and upgrade the existing System increasing its efficiency and effectiveness. The software improves the working methods replacing the existing manual system with computer based system.
* Chess is a two-player game which is played on a chessboard.
* Chess comprises on 64 squares arranged by eight rows and eight columns. It is very popular games played internationally by millions of people.
* This document explains all features, functions, and constraints of this program. This Chess game is built to allow remote users to play each other in chess.
* Its main focus is just to let people play; as users log into the system they are paired with the first available player and can proceed. The program needs to be intuitive, reliable, and easy to use.
* The scope of this project is to provide chess game that is intuitive and entertaining for players of all skill levels.
* The game also provides some useful features, such as the ability to save a game and return to it later. In general, it focuses on providing a simple, streamlined playing experience.
* Before going to what we have improved first let us know what is the existing system. That is as below.
* Here they need to play the game manually. Where two people play the chess game on the normal chess board.
* The online chess game apps provide timer for each move or timer together but they does not provide Timer for each player for the whole game so some people waste their time for manipulating the other player where as in original competitions the each player is provided with limited amount of time for the total game not for each move.
* So Time wastage is the major issue in Existing System.
* Here in this project we have modified it as below
* The main objective of Chess game is to enhance and upgrade the existing System increasing its efficiency and effectiveness. The software improves the working methods replacing the existing manual system with computer based system.
* Proposed system will give a new and bright future to the Chess game.
* Developing the chess game.
* By adding Timer to it in such a way that time is allocated individually to each player and time allocated is equal for both the players.
* If a player waste his time (when one player’s time reaches Zero) then automatically the opponent will be declared as the winner.

**3.REQUIREMENTS SPECIFICATIONS**

**Software Requirements:**

1. **Required For Running**

* Operating system: - Windows 10
* Environment: - Java Runtime Environment

1. **Required For Coding**

* IDE :Eclipse or NetBeans
* Java swing GUI libraries.

.

**HARDWARE REQUIREMENTS**

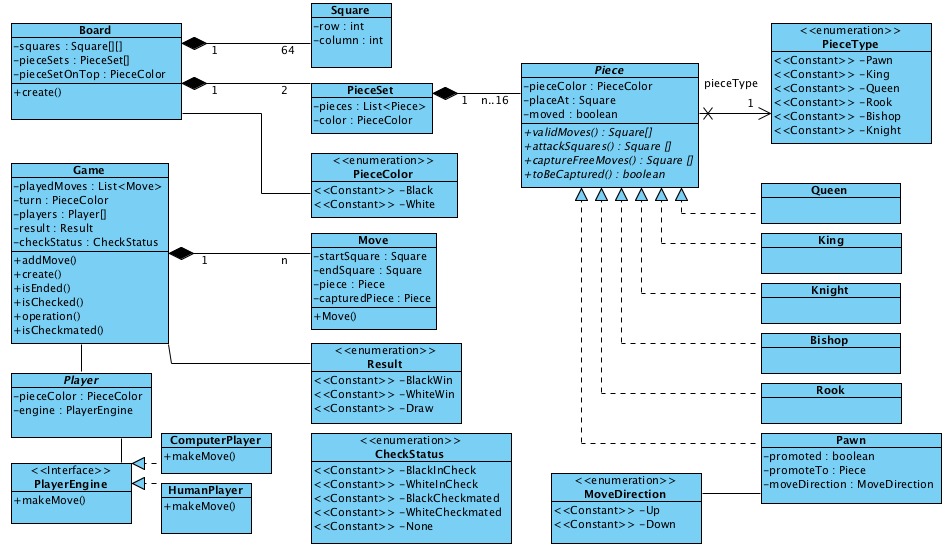
* System : Intel i3Core.5th gen
* Hard Disk : 80GB.
* Monitor : 14”LED
* Input Devices : Mouse
* Ram : 4GB

**4. SYSTEM DESIGN**

**System Architecture**

* System design is the process of applying various techniques and principles for the purpose of definition a system in sufficient details to permit its physical realization.
* Software design is the kernel of the software engineering process. Once the software
* Requirements have been analyzed and specified, the design is the first activity. The flow of information during this process is as follows.
* This game is built using core java, java swing GUI libraries and the Unit test suite.
* It uses custom drawing for game components and self-programmed logic for checkmate detection.
* The code is modular, standalone and object oriented

**Class Diagram:**



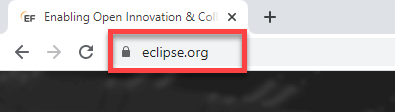
### 5. IMPLEMENTATION

**EnvironmentalSetup:**

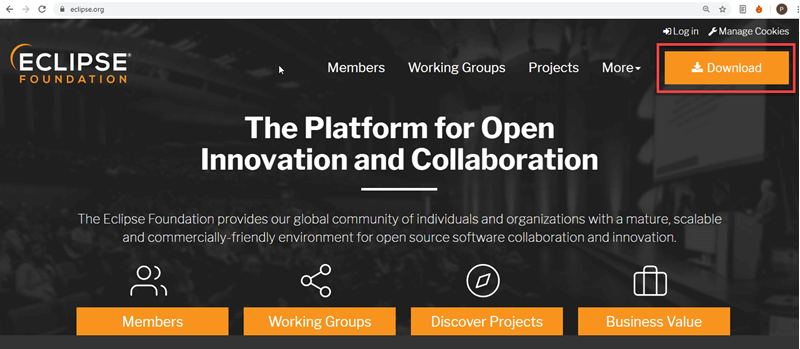
* we need to install and setup the IDE
* after installing we need to set the path in environmental variables
* the process for installing is as below

**Step 1)** Installing Eclipse

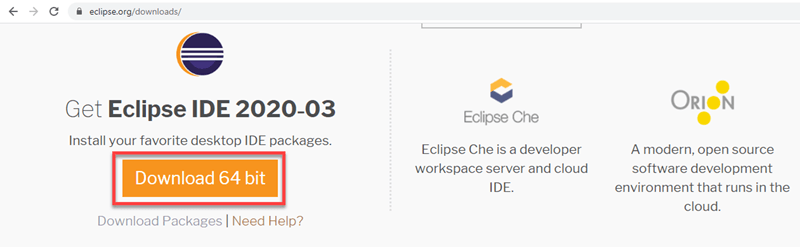
Open your browser and type <https://www.eclipse.org/>



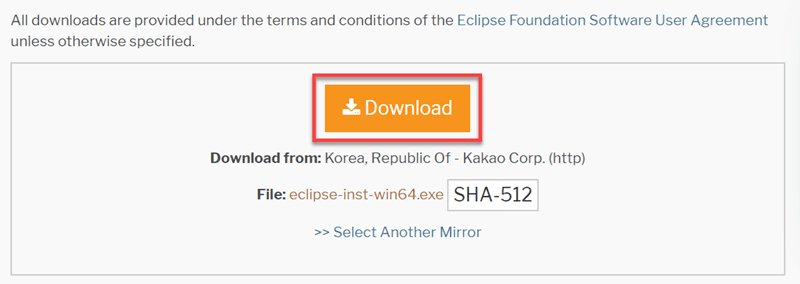
**Step 2)** Click on “Download” button.



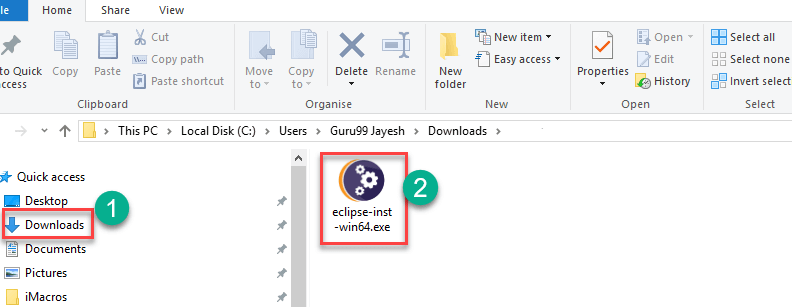
**Step 3)**Click on “Download 64 bit” button



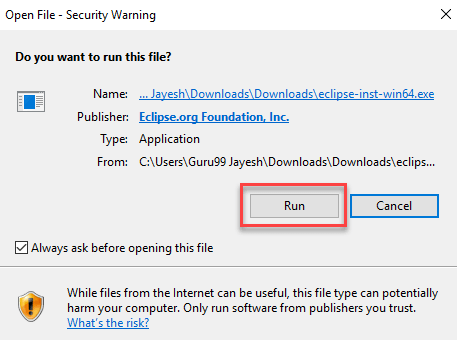
**Step 4)**Click on “Download” button



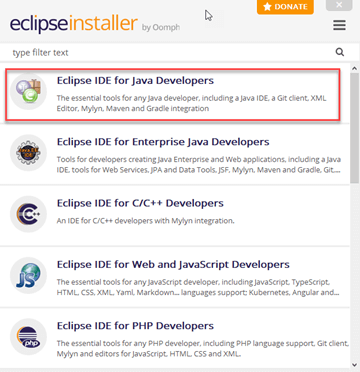
**Step 5)**Install Eclipse.

1. Click on “downloads” in Windows file explorer.
2. Click on “eclipse-inst-win64.exe” file.

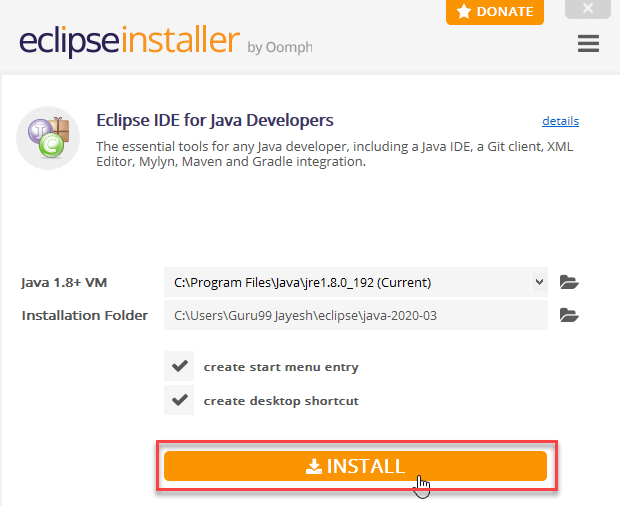
**Step 6)**Click on Run button

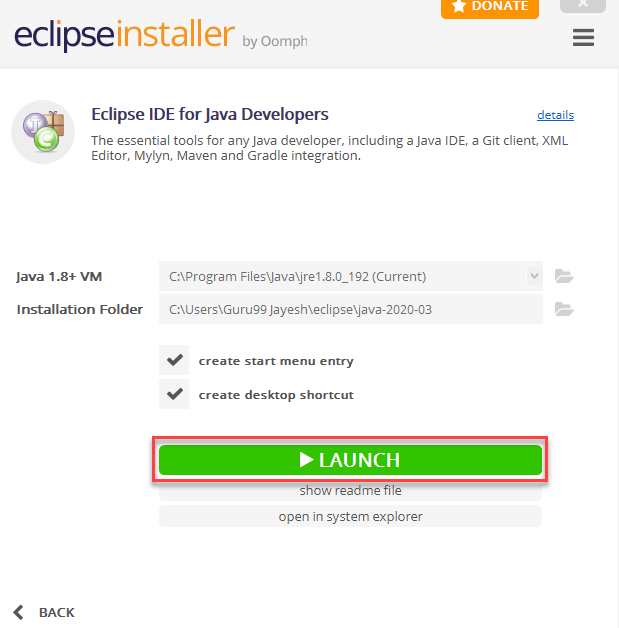


**Step 7)**Click on “Eclipse IDE for Java Developers”

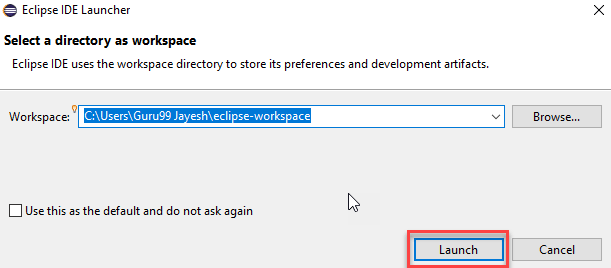


**Step 8)**Click on “INSTALL” button

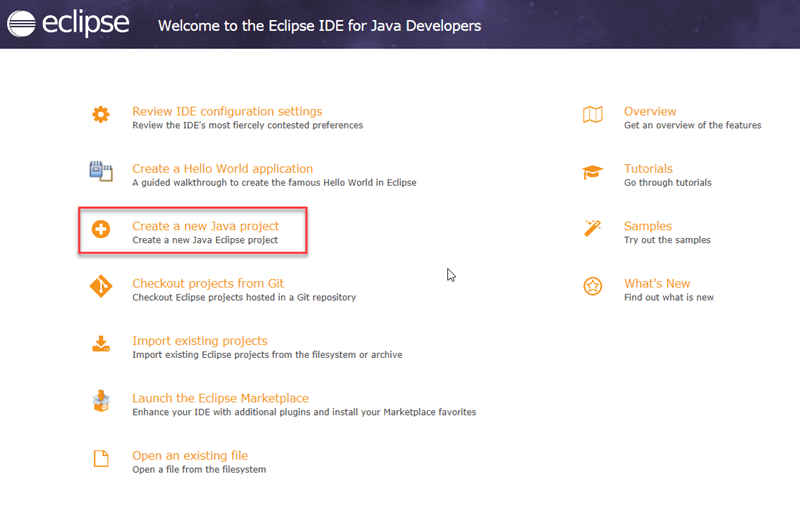


**Step 9)**Click on “LAUNCH” button.

**Step 10)**Click on “Launch” button.

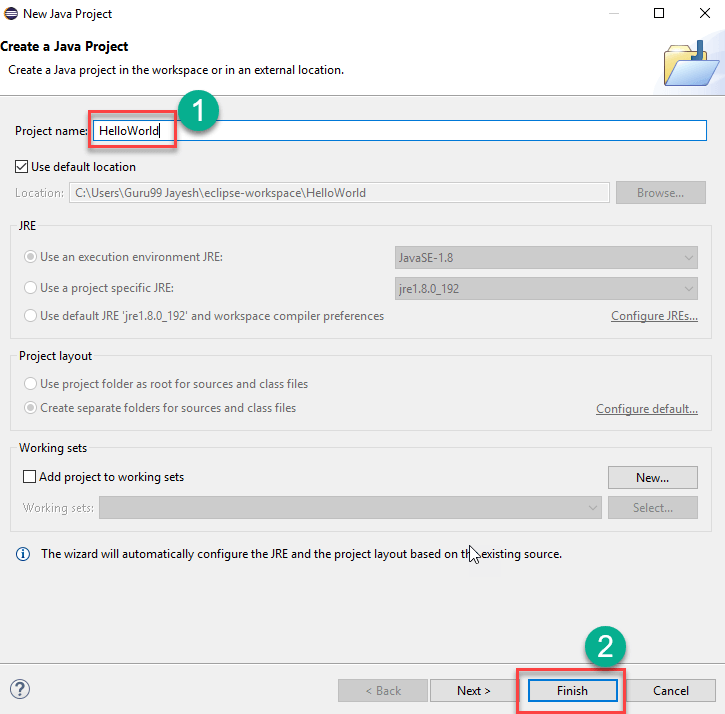


**Step 11)**Click on “Create a new Java project” link.



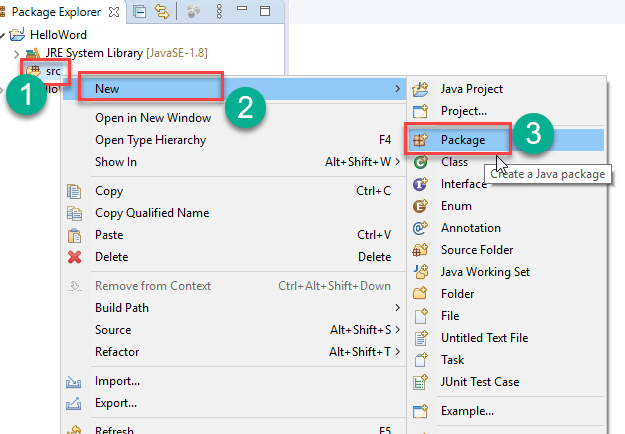
**Step 12)**Create a new Java Project

1. Write project name.
2. Click on “Finish button”.



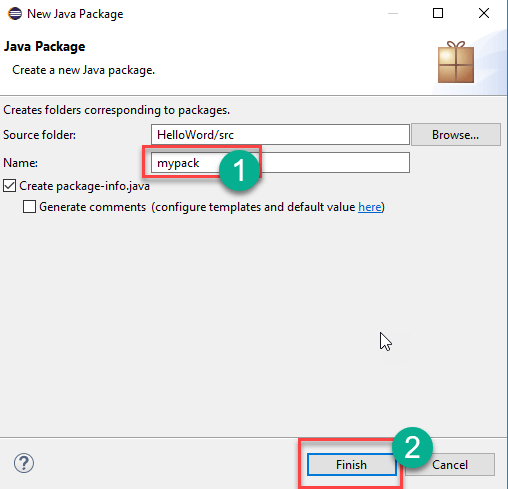
**Step 13)**Create Java Package.

1. Goto “src”.
2. Click on “New”.
3. Click on “Package”.



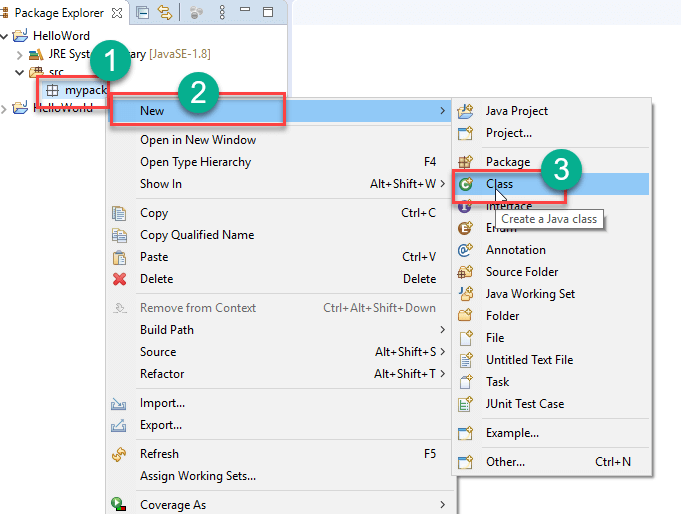
**Step 14)** Writing package name.

1. Write name of the package
2. Click on Finish button.



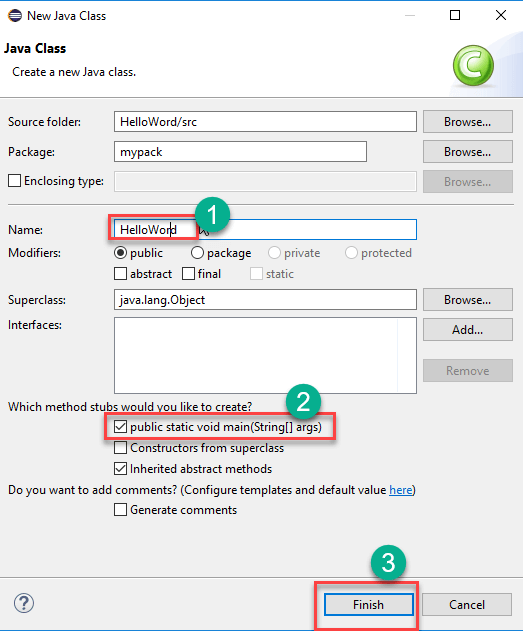
**Step 15)**Creating Java Class

1. Click on package you have created.
2. Click on “New”.
3. Click on “Class”.



**Step 16)**Defining Java Class.

1. Write class name
2. Click on “public static void main (String[] args)” checkbox.
3. Click on “Finish” button.



**Module Description:**

The idea is basically to build a chess game program that allows two players to play against one another as if they were playing a real board game without a computer.

The programming language used will be Java. Due to time constraints, which are set to 50 hours of work, the focus will be on programming the most essential components first that are needed for the program to work and anything else should be considered a bonus.

**SoftwareDescription:**

* This game is built using core java.
* Java swing GUI libraries and the Unit test suite.
* It uses custom drawing for game components and self-programmed logic for checkmate detection.

**SampleCode:**

* **GAME.java**

**package** chess;

**import** javax.swing.\*;

**publicclass** Game **implements** Runnable {

**publicvoid** run() {

SwingUtilities.*invokeLater*(**new** StartMenu());

}

**publicstaticvoid** main(String[] args) {

SwingUtilities.*invokeLater*(**new** Game());

}

}

* **StartMenu.java**

package chess;

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.net.URL;

import javax.imageio.ImageIO;

import javax.swing.Box;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class StartMenu implements Runnable {

public void run() {

final JFrame startWindow = new JFrame("Chess");

// Set window properties

startWindow.setLocation(300,100);

startWindow.setResizable(false);

startWindow.setSize(260, 240);

Box components = Box.createVerticalBox();

startWindow.add(components);

// Game title

final JPanel titlePanel = new JPanel();

components.add(titlePanel);

final JLabel titleLabel = new JLabel("Chess");

titlePanel.add(titleLabel);

// Black player selections

final JPanel blackPanel = new JPanel();

components.add(blackPanel, BorderLayout.EAST);

final JLabel blackPiece = new JLabel();

try {

Image blackImg = ImageIO.read(getClass().getResource("bp.png"));

blackPiece.setIcon(new ImageIcon(blackImg));

blackPanel.add(blackPiece);

} catch (Exception e) {

System.out.println("Required game file bp.png image is missing in the given path");

}

final JTextField blackInput = new JTextField("Black", 10);

blackPanel.add(blackInput);

// White player selections

final JPanel whitePanel = new JPanel();

components.add(whitePanel);

final JLabel whitePiece = new JLabel();

try {

Image whiteImg = ImageIO.read(getClass().getResource("wp.png"));

whitePiece.setIcon(new ImageIcon(whiteImg));

whitePanel.add(whitePiece);

startWindow.setIconImage(whiteImg);

} catch (Exception e) {

System.out.println("Required game file wp.png missing");

}

final JTextField whiteInput = new JTextField("White", 10);

whitePanel.add(whiteInput);

// Timer settings

final String[] minSecInts = new String[60];

for (int i = 0; i < 60; i++) {

if (i < 10) {

minSecInts[i] = "0" + Integer.toString(i);

} else {

minSecInts[i] = Integer.toString(i);

}

}

final JComboBox<String> seconds = new JComboBox<String>(minSecInts);

final JComboBox<String> minutes = new JComboBox<String>(minSecInts);

final JComboBox<String> hours =

new JComboBox<String>(new String[] {"0","1","2","3"});

Box timerSettings = Box.createHorizontalBox();

hours.setMaximumSize(hours.getPreferredSize());

minutes.setMaximumSize(minutes.getPreferredSize());

seconds.setMaximumSize(minutes.getPreferredSize());

timerSettings.add(hours);

timerSettings.add(Box.createHorizontalStrut(10));

timerSettings.add(seconds);

timerSettings.add(Box.createHorizontalStrut(10));

timerSettings.add(minutes);

timerSettings.add(Box.createVerticalGlue());

components.add(timerSettings);

// Buttons

Box buttons = Box.createHorizontalBox();

final JButton quit = new JButton("Quit");

quit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

startWindow.dispose();

}

});

final JButton instr = new JButton("Instructions");

instr.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(startWindow,

"To begin a new game, input player names\n" +

"next to the pieces. Set the clocks and\n" +

"click \"Start\". Setting the timer to all\n" +

"zeroes begins a new untimed game.",

"How to play",

JOptionPane.PLAIN\_MESSAGE);

}

});

final JButton start = new JButton("Start");

start.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String bn = blackInput.getText();

String wn = whiteInput.getText();

int hh = Integer.parseInt((String) hours.getSelectedItem());

int mm = Integer.parseInt((String) minutes.getSelectedItem());

int ss = Integer.parseInt((String) seconds.getSelectedItem());

new GameWindow(bn, wn, hh, mm, ss);

startWindow.dispose();

}

});

buttons.add(start);

buttons.add(Box.createHorizontalStrut(10));

buttons.add(instr);

buttons.add(Box.createHorizontalStrut(10));

buttons.add(quit);

components.add(buttons);

Component space = Box.createGlue();

components.add(space);

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

startWindow.setVisible(true);

}

}

* **GameWindow.java**

package chess;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import javax.imageio.ImageIO;

import javax.swing.\*;

public class GameWindow {

private JFrame gameWindow;

public Clock blackClock;

public Clock whiteClock;

private Timer timer;

private Board board;

public GameWindow(String blackName, String whiteName, int hh,

int mm, int ss) {

blackClock = new Clock(hh, ss, mm);

whiteClock = new Clock(hh, ss, mm);

gameWindow = new JFrame("Chess");

try {

Image whiteImg = ImageIO.read(getClass().getResource("wp.png"));

gameWindow.setIconImage(whiteImg);

} catch (Exception e) {

System.out.println("Game file wp.png not found");

}

gameWindow.setLocation(100, 100);

gameWindow.setLayout(new BorderLayout(20,20));

// Game Data window

JPanel gameData = gameDataPanel(blackName, whiteName, hh, mm, ss);

gameData.setSize(gameData.getPreferredSize());

gameWindow.add(gameData, BorderLayout.NORTH);

this.board = new Board(this);

gameWindow.add(board, BorderLayout.CENTER);

gameWindow.add(buttons(), BorderLayout.SOUTH);

gameWindow.setMinimumSize(gameWindow.getPreferredSize());

gameWindow.setSize(gameWindow.getPreferredSize());

gameWindow.setResizable(true);

gameWindow.pack();

gameWindow.setVisible(true);

gameWindow.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}

// Helper function to create data panel

private JPanel gameDataPanel(final String bn, final String wn, final int hh, final int mm, final int ss) {

JPanel gameData = new JPanel();

gameData.setLayout(new GridLayout(3,2,0,0));

// PLAYER NAMES

JLabel w = new JLabel(wn);

JLabel b = new JLabel(bn);

w.setHorizontalAlignment(JLabel.CENTER);

w.setVerticalAlignment(JLabel.CENTER);

b.setHorizontalAlignment(JLabel.CENTER);

b.setVerticalAlignment(JLabel.CENTER);

w.setSize(w.getMinimumSize());

b.setSize(b.getMinimumSize());

gameData.add(w);

gameData.add(b);

// CLOCKS

final JLabel bTime = new JLabel(blackClock.getTime());

final JLabel wTime = new JLabel(whiteClock.getTime());

bTime.setHorizontalAlignment(JLabel.CENTER);

bTime.setVerticalAlignment(JLabel.CENTER);

wTime.setHorizontalAlignment(JLabel.CENTER);

wTime.setVerticalAlignment(JLabel.CENTER);

if (!(hh == 0 && mm == 0 && ss == 0)) {

timer = new Timer(1000, null);

timer.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

boolean turn = board.getTurn();

if (turn) {

whiteClock.decr();

wTime.setText(whiteClock.getTime());

if (whiteClock.outOfTime()) {

timer.stop();

int n = JOptionPane.showConfirmDialog(

gameWindow,

bn + " wins by time! Play a new game? \n" +

"Choosing \"No\" quits the game.",

bn + " wins!",

JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

new GameWindow(bn, wn, hh, mm, ss);

gameWindow.dispose();

} else gameWindow.dispose();

}

} else {

blackClock.decr();

bTime.setText(blackClock.getTime());

if (blackClock.outOfTime()) {

timer.stop();

int n = JOptionPane.showConfirmDialog(

gameWindow,

wn + " wins by time! Play a new game? \n" +

"Choosing \"No\" quits the game.",

wn + " wins!",

JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

new GameWindow(bn, wn, hh, mm, ss);

gameWindow.dispose();

} else gameWindow.dispose();

}

}

}

});

timer.start();

} else {

wTime.setText("Untimed game");

bTime.setText("Untimed game");

}

gameData.add(wTime);

gameData.add(bTime);

gameData.setPreferredSize(gameData.getMinimumSize());

return gameData;

}

private JPanel buttons() {

JPanel buttons = new JPanel();

buttons.setLayout(new GridLayout(1, 3, 10, 0));

final JButton quit = new JButton("Quit");

quit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(

gameWindow,

"Are you sure you want to quit?",

"Confirm quit", JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

if (timer != null) timer.stop();

gameWindow.dispose();

}

}

});

final JButton nGame = new JButton("New game");

nGame.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int n = JOptionPane.showConfirmDialog(

gameWindow,

"Are you sure you want to begin a new game?",

"Confirm new game", JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

SwingUtilities.invokeLater(new StartMenu());

gameWindow.dispose();

}

}

});

final JButton instr = new JButton("How to play");

instr.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(gameWindow,

"Move the chess pieces on the board by clicking\n"

+ "and dragging. The game will watch out for illegal\n"

+ "moves. You can win either by your opponent running\n"

+ "out of time or by checkmating your opponent.\n"

+ "\nGood luck, hope you enjoy the game!",

"How to play",

JOptionPane.PLAIN\_MESSAGE);

}

});

buttons.add(instr);

buttons.add(nGame);

buttons.add(quit);

buttons.setPreferredSize(buttons.getMinimumSize());

return buttons;

}

public void checkmateOccurred (int c) {

if (c == 0) {

if (timer != null) timer.stop();

int n = JOptionPane.showConfirmDialog(

gameWindow,

"White wins by checkmate! Set up a new game? \n" +

"Choosing \"No\" lets you look at the final situation.",

"White wins!",

JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

SwingUtilities.invokeLater(new StartMenu());

gameWindow.dispose();

}

} else {

if (timer != null) timer.stop();

int n = JOptionPane.showConfirmDialog(

gameWindow,

"Black wins by checkmate! Set up a new game? \n" +

"Choosing \"No\" lets you look at the final situation.",

"Black wins!",

JOptionPane.YES\_NO\_OPTION);

if (n == JOptionPane.YES\_OPTION) {

SwingUtilities.invokeLater(new StartMenu());

gameWindow.dispose();

}

}

}

}

* **Board.java**

package chess;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.GridLayout;

import java.awt.Image;

import java.awt.Point;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import java.util.LinkedList;

import java.util.List;

import javax.swing.\*;

@SuppressWarnings("serial")

public class Board extends JPanel implements MouseListener, MouseMotionListener {

// Resource location constants for piece images

private static final String RESOURCES\_WBISHOP\_PNG = "wbishop.png";

private static final String RESOURCES\_BBISHOP\_PNG = "bbishop.png";

private static final String RESOURCES\_WKNIGHT\_PNG = "wknight.png";

private static final String RESOURCES\_BKNIGHT\_PNG = "bknight.png";

private static final String RESOURCES\_WROOK\_PNG = "wrook.png";

private static final String RESOURCES\_BROOK\_PNG = "brook.png";

private static final String RESOURCES\_WKING\_PNG = "wking.png";

private static final String RESOURCES\_BKING\_PNG = "bking.png";

private static final String RESOURCES\_BQUEEN\_PNG = "bqueen.png";

private static final String RESOURCES\_WQUEEN\_PNG = "wqueen.png";

private static final String RESOURCES\_WPAWN\_PNG = "wpawn.png";

private static final String RESOURCES\_BPAWN\_PNG = "bpawn.png";

// Logical and graphical representations of board

private final Square[][] board;

private final GameWindow g;

// List of pieces and whether they are movable

public final LinkedList<Piece> Bpieces;

public final LinkedList<Piece> Wpieces;

public List<Square> movable;

private boolean whiteTurn;

private Piece currPiece;

private int currX;

private int currY;

private CheckmateDetector cmd;

public Board(GameWindow g) {

this.g = g;

board = new Square[8][8];

Bpieces = new LinkedList<Piece>();

Wpieces = new LinkedList<Piece>();

setLayout(new GridLayout(8, 8, 0, 0));

this.addMouseListener(this);

this.addMouseMotionListener(this);

for (int x = 0; x < 8; x++) {

for (int y = 0; y < 8; y++) {

int xMod = x % 2;

int yMod = y % 2;

if ((xMod == 0 && yMod == 0) || (xMod == 1 && yMod == 1)) {

board[x][y] = new Square(this, 1, y, x);

this.add(board[x][y]);

} else {

board[x][y] = new Square(this, 0, y, x);

this.add(board[x][y]);

}

}

}

initializePieces();

this.setPreferredSize(new Dimension(400, 400));

this.setMaximumSize(new Dimension(400, 400));

this.setMinimumSize(this.getPreferredSize());

this.setSize(new Dimension(400, 400));

whiteTurn = true;

}

private void initializePieces() {

for (int x = 0; x < 8; x++) {

board[1][x].put(new Pawn(0, board[1][x], RESOURCES\_BPAWN\_PNG));

board[6][x].put(new Pawn(1, board[6][x], RESOURCES\_WPAWN\_PNG));

}

board[7][3].put(new Queen(1, board[7][3], RESOURCES\_WQUEEN\_PNG));

board[0][3].put(new Queen(0, board[0][3], RESOURCES\_BQUEEN\_PNG));

King bk = new King(0, board[0][4], RESOURCES\_BKING\_PNG);

King wk = new King(1, board[7][4], RESOURCES\_WKING\_PNG);

board[0][4].put(bk);

board[7][4].put(wk);

board[0][0].put(new Rook(0, board[0][0], RESOURCES\_BROOK\_PNG));

board[0][7].put(new Rook(0, board[0][7], RESOURCES\_BROOK\_PNG));

board[7][0].put(new Rook(1, board[7][0], RESOURCES\_WROOK\_PNG));

board[7][7].put(new Rook(1, board[7][7], RESOURCES\_WROOK\_PNG));

board[0][1].put(new Knight(0, board[0][1], RESOURCES\_BKNIGHT\_PNG));

board[0][6].put(new Knight(0, board[0][6], RESOURCES\_BKNIGHT\_PNG));

board[7][1].put(new Knight(1, board[7][1], RESOURCES\_WKNIGHT\_PNG));

board[7][6].put(new Knight(1, board[7][6], RESOURCES\_WKNIGHT\_PNG));

board[0][2].put(new Bishop(0, board[0][2], RESOURCES\_BBISHOP\_PNG));

board[0][5].put(new Bishop(0, board[0][5], RESOURCES\_BBISHOP\_PNG));

board[7][2].put(new Bishop(1, board[7][2], RESOURCES\_WBISHOP\_PNG));

board[7][5].put(new Bishop(1, board[7][5], RESOURCES\_WBISHOP\_PNG));

for(int y = 0; y < 2; y++) {

for (int x = 0; x < 8; x++) {

Bpieces.add(board[y][x].getOccupyingPiece());

Wpieces.add(board[7-y][x].getOccupyingPiece());

}

}

cmd = new CheckmateDetector(this, Wpieces, Bpieces, wk, bk);

}

public Square[][] getSquareArray() {

return this.board;

}

public boolean getTurn() {

return whiteTurn;

}

public void setCurrPiece(Piece p) {

this.currPiece = p;

}

public Piece getCurrPiece() {

return this.currPiece;

}

@Override

public void paintComponent(Graphics g) {

// super.paintComponent(g);

for (int x = 0; x < 8; x++) {

for (int y = 0; y < 8; y++) {

Square sq = board[y][x];

sq.paintComponent(g);

}

}

if (currPiece != null) {

if ((currPiece.getColor() == 1 && whiteTurn)

|| (currPiece.getColor() == 0 && !whiteTurn)) {

final Image i = currPiece.getImage();

g.drawImage(i, currX, currY, null);

}

}

}

@Override

public void mousePressed(MouseEvent e) {

currX = e.getX();

currY = e.getY();

Square sq = (Square) this.getComponentAt(new Point(e.getX(), e.getY()));

if (sq.isOccupied()) {

currPiece = sq.getOccupyingPiece();

if (currPiece.getColor() == 0 && whiteTurn)

return;

if (currPiece.getColor() == 1 && !whiteTurn)

return;

sq.setDisplay(false);

}

repaint();

}

@Override

public void mouseReleased(MouseEvent e) {

Square sq = (Square) this.getComponentAt(new Point(e.getX(), e.getY()));

if (currPiece != null) {

if (currPiece.getColor() == 0 && whiteTurn)

return;

if (currPiece.getColor() == 1 && !whiteTurn)

return;

List<Square> legalMoves = currPiece.getLegalMoves(this);

movable = cmd.getAllowableSquares(whiteTurn);

if (legalMoves.contains(sq) && movable.contains(sq)

&&cmd.testMove(currPiece, sq)) {

sq.setDisplay(true);

currPiece.move(sq);

cmd.update();

if (cmd.blackCheckMated()) {

currPiece = null;

repaint();

this.removeMouseListener(this);

this.removeMouseMotionListener(this);

g.checkmateOccurred(0);

} else if (cmd.whiteCheckMated()) {

currPiece = null;

repaint();

this.removeMouseListener(this);

this.removeMouseMotionListener(this);

g.checkmateOccurred(1);

}

else {

currPiece = null;

whiteTurn = !whiteTurn;

movable = cmd.getAllowableSquares(whiteTurn);

}

} else {

currPiece.getPosition().setDisplay(true);

currPiece = null;

}

}

repaint();

}

@Override

public void mouseDragged(MouseEvent e) {

currX = e.getX() - 24;

currY = e.getY() - 24;

repaint();

}

// Irrelevant methods, do nothing for these mouse behaviors

@Override

public void mouseMoved(MouseEvent e) {

}

@Override

public void mouseClicked(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}

### 6. SYSTEMTESTING

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system correct the results will be successfully achieved. Effective testing early in the process translates directly into long term cost saving, reduced number of errors. System testing is done when all the modules of the system are in working order and has been tested independently for proper working. All the pieces are put into one system and test to determined, whether it needs user's requirements. The best program is worthless if doesn't needs. System testing is designed to uncover weakness that were not found in earlier tests like program testing in which only syntactical and logical are removed. The purpose of System Testing is to consider all the likely variations to which it will be subjected and then push the system to its limits.

A unit test should be very simple and test a function that does 1 single thing. Then you can combine the functions with some reasonable expectation that they will work. For example, I would expect a unit test for each piece to determine if a particular move is legal. A unit test for each piece to determine if it is putting the king in check. A test for each piece to determine where it is attacking, etc.

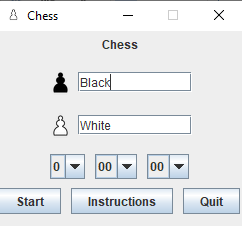
Testing a position seems like a very complicated unit test and would be much harder to do thoroughly. Instead write smaller tests against smaller functions and then know that those individually work - evaluating a position is just a matter of iterating over the simple functions.

If you want to test a position for a good (not forced) move, I think unit test's will artificially limit the long term development and strength of your chess engine... a binary result of a unit test will force your engine to make the same move every time.

I'd also look at adding unit tests for 'most direct' path to a mate with known endgames. I'd look to add unit tests for traversing through known openings as well. Mid game unit tests will be much harder - maybe plugging in a position and evaluating that the engine produces a usable result (which is a binary response).

**7. RESULT SCREENSHOTS**

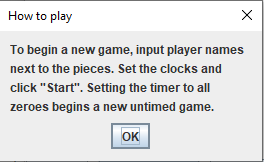
* Window for setting the player names and time.



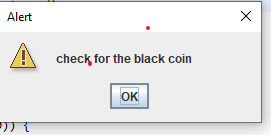
* The chess board after starting game.



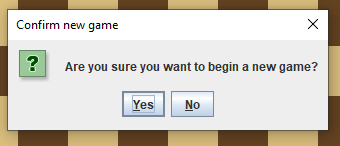
* Instructions

****

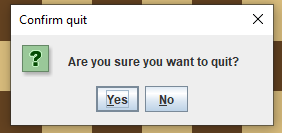
* Warning message

****

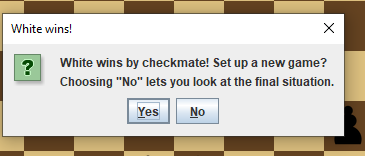
* New Game

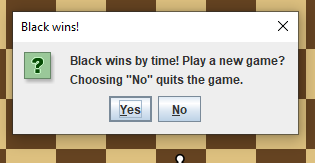
****

* Quit game

****

* Game Over messages.

****

****

**8. CONCLUSION**

This project aimed to create a practical and usable product, which can be considered as software and even an educational tool. A real life situation, which is a chess game in the current case, was modeled without any restrictions and based on its models personal understanding. Another aim of this Final Project was to analyze two agent base approaches. It was a fascinating a quite helpful experience for us to observe the differences between the two approaches by producing a practical work rather than conducting just a theoretical research. The users will observe and realize the fact that it is a very open-ended model, allowing users to interact with it using their own imagination, which was the main idea behind all this effort.

Chess which do play between two players on a board is intellectual and mental game, it has its own rules of play which help to enhance and improve the mental and intellectual activities of the player, and this game has a huge amount of players around the all world they have strongly interested to have play it. This document deals with the fully computerized Chess Game, first, the game computerizes for two player to do play chess according all the valid rules of the chess on computer.

The Game is won when an opponent’s king cannot move anywhere without being captured or if the time of a player exceeds.

**9. BIBLIOGRAPHY:-**

**Some of the best chess books ever written**

There are thousands of books available on chess, covering just about every aspect you can imagine. Listed here are a few titles considered classics by many experts on the subject.

Classic chess books frequently go in and out of print. If you can't find a new copy of a title that interests you, check out used book resources. And don't forget your local library!

**Chess for Idle Moments**

Looking for a single volume that will tell you everything you need to know about chess? These books are great for vacation or bedtime reading.

**The Immortal Game: A History of Chess by David Shenk.** This book won't teach you much about playing the game, but it will explain why chess is so popular. If you take it on vacation, you won't even need a chess set.

**The Mammoth Book of Chess by Graham Burgess**

• Over 500 pages cover all the technical aspects of chess: tactics, openings, glossary, and much more. If you take it on vacation, you will need a chess set to take full advantage of it.

**Chess Manuals**

These two books, written in the first half of the 20th century, cover all aspects of the game, starting with how the pieces move through example games. The opening sections are out-of-date, but this is true of any manual -- opening theory moves too fast.

**Lasker's Manual of Chess by Emanuel Lasker**

• Lasker was World Champion from 1894 to 1921 and an excellent writer. The chapter on position play is particularly good. Lasker was also a philosopher and shares his thinking throughout the book.

**The Game of Chess by Siegbert Tarrasch.**

Tarrasch, Lasker's nemesis at the turn of the last century, was one of the greatest early teachers of the game. The chapters on the endgame and the middlegame are good introductions to the subject

**GITHUB link**

. https://github.com/DhanaLakshmi2000/chess