# BÀI THI KẾT THÚC HỌC PHẦN

MÔN: LẬP TRÌNH MẠNG

HỌ TÊN SV: NGUYỄN BÁ TỚI MSV: 08D4800061

SBD: 26

PHÒNG THI SỐ: 2

```
package MangMayTinh.Chess.Connection.Server;

import MangMayTinh.Chess.Model.Enum.MessageType;
import MangMayTinh.Chess.Model.Interface.PlayerInterface;
import MangMayTinh.Chess.Model.Move;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Ba Toi
 */
public class Player extends Thread {

    Socket socket;
```

```
    ObjectOutputStream sender;
    ObjectInputStream receiver;
    boolean isFirstPlayer = false;
    PlayerInterface delegate;
    boolean isReady = false;
    boolean isRunning = false;


    public Player(Socket socket, ObjectOutputStream sender, ObjectInputStream
receiver) {
        this.socket = socket;
        this.sender = sender;
        this.receiver = receiver;
    }


    @Override
    public void run() {
        this.isRunning = true;
        try {
            while (isRunning) {
                MessageType type = (MessageType) receiver.readObject();
                switch (type) {
                    case move:
                        Move move = (Move) receiver.readObject();
                        if (this.delegate != null) {
```

```
                    System.out.println("Move got: " + move.getSource().x + " " +
move.getSource().y + " to : " + move.getDestination().x + " " +
move.getDestination().y);
                    Move newMove = move.clone();
                    this.delegate.move(newMove, this.isFirstPlayer);
                } else {
                    System.out.println("Can not move, please set delegate for
player!");
                }
                break;
            case message:
                String message = (String) receiver.readObject();
                if (this.delegate != null) {
                    this.delegate.didReceiveMessage(message, this.isFirstPlayer);
                } else {
                    System.out.println("Can not message, please set delegate for
player!");
                }
                break;
            case name:
                String name = (String) receiver.readObject();
                if (this.delegate != null) {
                    this.delegate.setName(name, this.isFirstPlayer);
                } else {
                    System.out.println("Can not set name, please set delegate for
player!");
                }
```

```
                    break;
                case surrender:
                    if (this.delegate != null) {
                        this.delegate.surrender(this.isFirstPlayer);
                    } else {
                        System.out.println("Can not set name, please set delegate for
player!");
                    }
                    break;
                case endGame:
                    this.isRunning = false;
                    if (this.delegate != null) {
                        this.delegate.endGame(isFirstPlayer);
                    } else {
                        System.out.println("Can not end game, please set delegate for
player!");
                    }
                    break;
                default:
                    System.out.println("Can not cast data from socket to expect
message type!");
                    break;
            }
        }
    } catch (IOException ex) {
        Logger.getLogger(Player.class.getName()).log(Level.SEVERE, null, ex);
```

```
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Player.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public <T> void sendMessage(MessageType type, T data) {
    try {
        this.sender.writeObject(type);
        this.sender.writeObject(data);
    } catch (Exception e) {
        System.out.print("Send data Error: ");
        System.out.println(e.toString());
    }
}

public void sendOperation(MessageType type) {
    try {
        this.sender.writeObject(type);
    } catch (Exception e) {
        System.out.print("Send data Error: ");
        System.out.println(e.toString());
    }
}

public boolean isReady() {
```

```
    return isReady;
}

public void setIsReady(boolean isReady) {
    this.isReady = isReady;
}

public void setDelegate(PlayerInterface delegate) {
    this.delegate = delegate;
}

public boolean isFirstPlayer() {
    return isFirstPlayer;
}

public void setIsFirstPlayer(boolean isFirstPlayer) {
    this.isFirstPlayer = isFirstPlayer;
}

public Socket getSocket() {
    return socket;
}

public void setSocket(Socket socket) {
    this.socket = socket;
```

```
    }

    public ObjectOutputStream getSender() {
        return sender;
    }

    public void setSender(ObjectOutputStream sender) {
        this.sender = sender;
    }

    public ObjectInputStream getReceiver() {
        return receiver;
    }

    public void setReceiver(ObjectInputStream receiver) {
        this.receiver = receiver;
    }
}
```