[ 고급JAVA 프로그래밍 14주차 보고서 ]



21.12.12

1826074 오현진

1번

문제 : 계속해서 쫓아오는 괴물을 피하는 Thread 구현

코드 :

**package** application;

**import** javafx.application.\*;

**import** javafx.event.\*;

**import** javafx.scene.\*;

**import** javafx.scene.control.\*;

**import** javafx.scene.input.\*;

**import** javafx.scene.layout.\*;

**import** javafx.scene.paint.\*;

**import** javafx.stage.\*;

**class** GamePane **extends** AnchorPane

{

**static** **final** **long** ***monterDelay*** = 100;

**final** **int** AVATAR\_MOVE = 10;

Stage stage;

Label avatar;

Label monster;

**public** GamePane(Stage stage)

{

**this**.stage = stage;

avatar = **new** Label("@");

monster = **new** Label("M");

avatar.setPrefSize(15, 15);

avatar.setTextFill(Color.***RED***);

avatar.setLayoutX(140);

avatar.setLayoutY(140);

monster.setPrefSize(15, 15);

monster.setLayoutX(250);

monster.setLayoutY(250);

**this**.getChildren().addAll(avatar, monster);

**this**.setOnKeyPressed(**new** EventHandler<KeyEvent>() {

@Override

**public** **void** handle(KeyEvent e)

{

**if**(e.getCharacter() == "q")

System.*exit*(0);

**switch**(e.getCode())

{

**case** ***UP***:

avatar.setLayoutY(avatar.getLayoutY() - AVATAR\_MOVE);

**break**;

**case** ***DOWN***:

avatar.setLayoutY(avatar.getLayoutY() + AVATAR\_MOVE);

**break**;

**case** ***LEFT***:

avatar.setLayoutX(avatar.getLayoutX() - AVATAR\_MOVE);

**break**;

**case** ***RIGHT***:

avatar.setLayoutX(avatar.getLayoutX() + AVATAR\_MOVE);

**break**;

}

}

});

Thread th = **new** Thread(**new** MonsterThread(stage, monster, avatar, ***monterDelay***));

th.setDaemon(**true**);

th.start();

}

**class** MonsterThread **implements** Runnable

{

Stage stage;

Label from;

Label to;

**long** monsterDelay;

**final** **int** MONSTER\_MOVE = 5;

**public** MonsterThread(Stage stage, Label from, Label to, **long** monsterDelay)

{

**this**.stage = stage;

**this**.from = from;

**this**.to = to;

**this**.monsterDelay = monsterDelay;

}

@Override

**public** **void** run()

{

**int** x=(**int**)from.getLayoutX();

**int** y=(**int**)from.getLayoutY();

**while**(**true**)

{

**if**(to.getLayoutX() < from.getLayoutX())

x = x - MONSTER\_MOVE;

**else**

x = x + MONSTER\_MOVE;

**if**(to.getLayoutY() < from.getLayoutY())

y = y - MONSTER\_MOVE;

**else**

y = y + MONSTER\_MOVE;

from.setLayoutX(x);

from.setLayoutY(y);

**double** gapX = Math.*abs*(to.getLayoutX() - from.getLayoutX());

**double** gapY = Math.*abs*(to.getLayoutY() - from.getLayoutY());

**if**(gapX<10 && gapY<10)

Platform.*runLater*(()->{

stage.setTitle("Avata Dead!!");

});

**else**

Platform.*runLater*(()->{

stage.setTitle("Avata Alived!!");

});

**try** { Thread.*sleep*(monsterDelay); }

**catch**(InterruptedException e) { **return**; }

}

}

}

}

**public** **class** Ex1 **extends** Application {

@Override

**public** **void** start(Stage primaryStage) **throws** Exception

{

primaryStage.setTitle("Avata Alived!!");

GamePane root = **new** GamePane(primaryStage);

Scene scene = **new** Scene(root,300,300);

primaryStage.setScene(scene);

primaryStage.show();

root.requestFocus();

}

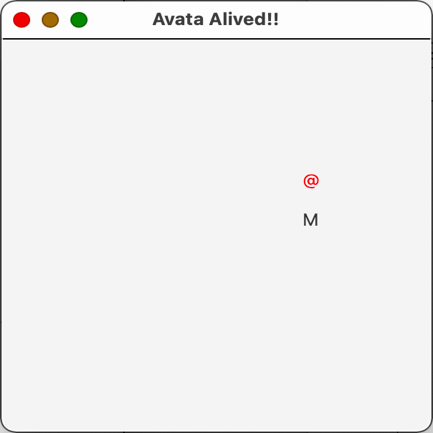
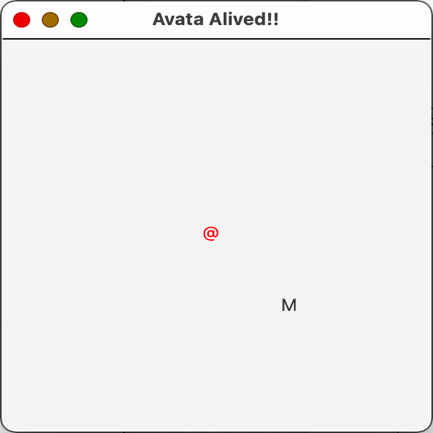
**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

실행 화면 :



2번

문제 : 총알을 발사해서 목표물을 맞추는 Thread 구현

코드 :

**package** application;

**import** java.applet.\*;

**import** java.net.\*;

**import** javafx.application.\*;

**import** javafx.event.\*;

**import** javafx.scene.\*;

**import** javafx.scene.control.\*;

**import** javafx.scene.image.\*;

**import** javafx.scene.input.\*;

**import** javafx.scene.layout.\*;

**import** javafx.scene.paint.\*;

**import** javafx.scene.text.\*;

**import** javafx.stage.\*;

**class** gp **extends** AnchorPane

{

TargetThread targetTh = **null**;

BulletThread bulletTh = **null**;

Label base = **new** Label();

Label bullet = **new** Label();

Label target = **new** Label();

URL resource1 = getClass().getResource("laser.wav");

URL resource2 = getClass().getResource("explosion.wav");

~~AudioClip~~ laserSound = ~~Applet~~.~~newAudioClip~~(resource1);

~~AudioClip~~ explosionSound = ~~Applet~~.~~newAudioClip~~(resource2);

**public** gp()

{

base.setPrefSize(40, 40);

base.setStyle("-fx-background-color: Black");

Image image = **new** Image(**this**.getClass()

.getResourceAsStream("spaceship.png"));

target.setGraphic(**new** ImageView(image));

target.setPrefSize(image.getWidth(), image.getHeight());

bullet.setFont(**new** Font(5));

bullet.setPrefSize(10, 10);

bullet.setStyle("-fx-background-color: Red");

**this**.getChildren().addAll(base, bullet, target);

}

**public** **void** startGame()

{

base.setLayoutX(**this**.getWidth()/2-20);

base.setLayoutY(**this**.getHeight()-40);

bullet.setLayoutX(**this**.getWidth()/2-5);

bullet.setLayoutY(**this**.getHeight()-50);

target.setLayoutX(0);

target.setLayoutY(0);

targetTh = **new** TargetThread();

targetTh.setDaemon(**true**);

targetTh.start();

base.requestFocus();

base.setOnKeyPressed(event->{

**if**(event.getCode() == KeyCode.***ENTER***)

{

**if**(bulletTh==**null** || !bulletTh.isAlive())

{

bulletTh = **new** BulletThread();

bulletTh.setDaemon(**true**);

bulletTh.start();

laserSound.~~play~~();

}

}

});

}

**class** TargetThread **extends** Thread

{

**public** **void** run() {

**while**(**true**) {

**double** x = target.getLayoutX() + 5;

**double** y = target.getLayoutY();

**if**(x > gp.**this**.getWidth())

{

Platform.*runLater*(()->{

target.setLayoutX(0);

target.setLayoutY(0);

});

} **else** {

Platform.*runLater*(()->{

target.setLayoutX(x);

target.setLayoutY(y);

});

}

**try** {

Thread.*sleep*(20);

} **catch**(InterruptedException e) {

Platform.*runLater*(()->{

target.setLayoutX(0);

target.setLayoutY(0);

});

**try** { Thread.*sleep*(1000); }

**catch**(InterruptedException ex) { }

}

}

}

}

**class** BulletThread **extends** Thread

{

@Override

**public** **void** run() {

**while**(**true**) {

**if**(hit())

{

Platform.*runLater*(()->{

targetTh.interrupt();

bullet.setLayoutX(gp.**this**.getWidth()/2-5);

bullet.setLayoutY(gp.**this**.getHeight()-50);

});

explosionSound.~~play~~();

**return**;

}

**else**

{

**double** x = bullet.getLayoutX();

**double** y = bullet.getLayoutY() - 5;

**if**(y < 0)

{

Platform.*runLater*(()->{

bullet.setLayoutX(gp.**this**.getWidth()/2-5);

bullet.setLayoutY(gp.**this**.getHeight()-50);

});

**return**;

}

Platform.*runLater*(()->{

bullet.setLayoutX(x);

bullet.setLayoutY(y);

});

}

**try** { *sleep*(20); }

**catch**(InterruptedException e) { }

}

}

**private** **boolean** hit()

{

**double** bx = bullet.getLayoutX() + bullet.getWidth() / 2;

**double** by = bullet.getLayoutY() + bullet.getHeight() / 2;

**double** tx = target.getLayoutX();

**double** ty = target.getLayoutY();

**double** tWidth = target.getWidth();

**double** tHeight = target.getHeight();

**if**(((tx <= bx) && (bx <= tx+tWidth-1)) &&

((ty <= by) && (by <= ty+tHeight-1)))

**return** **true**;

**else**

**return** **false**;

}

}

}

**public** **class** Ex2 **extends** Application {

**public** **void** start(Stage primaryStage) **throws** Exception

{

primaryStage.setTitle("Shooting Game");

gp root = **new** gp();

Scene scene = **new** Scene(root,300,300);

primaryStage.setScene(scene);

primaryStage.setResizable(**false**);

primaryStage.show();

root.startGame();

}

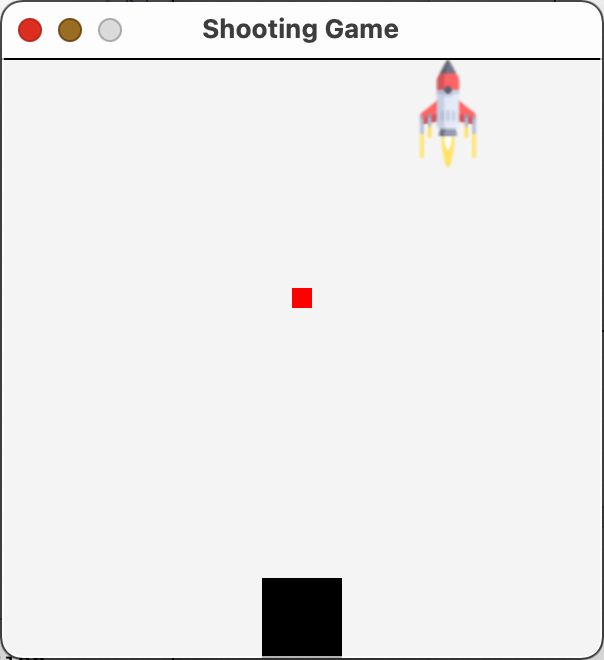
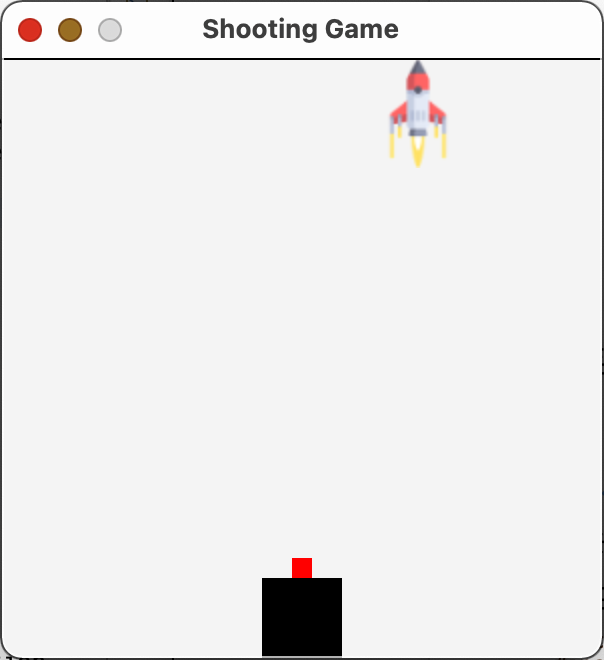
**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

실행 화면 :



3번

문제 : 채팅을 할 수 있는 Thread 구현

코드 :

<서버>

**package** application;

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**import** java.util.concurrent.\*;

**import** javafx.application.\*;

**import** javafx.event.\*;

**import** javafx.geometry.\*;

**import** javafx.scene.\*;

**import** javafx.scene.control.\*;

**import** javafx.scene.input.\*;

**import** javafx.scene.layout.\*;

**import** javafx.scene.paint.\*;

**import** javafx.scene.text.\*;

**import** javafx.stage.\*;

**public** **class** Ex3 **extends** Application {

ExecutorService threadPool;

ServerSocket serverSocket;

Vector<Client> clients = **new** Vector<Client>();

**class** Client {

Socket socket;

**public** Client(Socket socket)

{

**this**.socket = socket;

receive();

}

**public** **void** receive()

{

Runnable thread = **new** Runnable()

{

@Override

**public** **void** run()

{

**try**

{

**while**(**true**)

{

InputStream in = socket.getInputStream();

**byte**[] buf = **new** **byte**[100];

**int** len = in.read(buf);

**if**(len == -1)

**throw** **new** IOException();

System.***out***.println("[Success] "

+ socket.getRemoteSocketAddress()

+ ": " + Thread.*currentThread*().getName());

String msg = **new** String(buf, 0, len, "UTF-8");

**for**(Client client : clients)

{

client.send(msg);

}

}

}

**catch**(Exception e)

{

**try**

{

System.***out***.println("[Failure] "

+ socket.getRemoteSocketAddress()

+ ": " + Thread.*currentThread*().getName());

clients.remove(Client.**this**);

socket.close();

}

**catch** (Exception e2) { }

}

}

};

threadPool.submit(thread);

}

**public** **void** send(String msg)

{

Runnable thread = **new** Runnable()

{

@Override

**public** **void** run()

{

**try**

{

OutputStream out = socket.getOutputStream();

**byte**[] buf = msg.getBytes("UTF-8");

out.write(buf);

out.flush();

}

**catch**(Exception e)

{

**try**

{

System.***out***.println("[Failure] "

+ socket.getRemoteSocketAddress()

+ ": " + Thread.*currentThread*().getName());

clients.remove(Client.**this**);

socket.close();

}

**catch** (Exception e2) { }

}

}

};

threadPool.submit(thread);

}

}

**public** **void** startServer(String ipAddress, **int** port)

{

**try**

{

serverSocket = **new** ServerSocket();

serverSocket.bind(**new** InetSocketAddress(ipAddress, port));

}

**catch**(Exception e)

{

**if**(!serverSocket.isClosed())

stopServer();

**return**;

}

Runnable thread = **new** Runnable()

{

@Override

**public** **void** run()

{

**while**(**true**)

{

**try**

{

Socket socket = serverSocket.accept();

clients.add(**new** Client(socket));

System.***out***.println("[Connected with client] "

+ socket.getRemoteSocketAddress()

+ ": " + Thread.*currentThread*().getName());

}

**catch**(Exception e)

{

**if**(!serverSocket.isClosed())

stopServer();

**break**;

}

}

}

};

threadPool = Executors.*newCachedThreadPool*();

threadPool.submit(thread);

}

**public** **void** stopServer()

{

**try**

{

Iterator<Client> iterator = clients.iterator();

**while**(iterator.hasNext())

{

Client client = iterator.next();

client.socket.close();

iterator.remove();

}

**if**(serverSocket != **null** && !serverSocket.isClosed())

serverSocket.close();

**if**(threadPool != **null** && !threadPool.isShutdown())

threadPool.shutdown();

}

**catch**(Exception e) { }

}

**public** **void** start(Stage primaryStage) **throws** Exception

{

BorderPane root = **new** BorderPane();

root.setPadding(**new** Insets(5));

TextArea txtDisplay = **new** TextArea();

txtDisplay.setEditable(**false**);

txtDisplay.setFont(**new** Font("Consolas", 14));

root.setCenter(txtDisplay);

Button btnStartStop = **new** Button("Start");

btnStartStop.setMaxWidth(Double.***MAX\_VALUE***);

BorderPane.*setMargin*(btnStartStop, **new** Insets(1,0,0,0));

root.setBottom(btnStartStop);

String ipAddress = "localhost";

**int** port = 4000;

btnStartStop.setOnAction(event -> {

**if**(btnStartStop.getText().equals("Start"))

{

startServer(ipAddress, port);

Platform.*runLater*(()->{

String str = String.*format*("[Server(%s:%d) started...]\n",

ipAddress, port);

txtDisplay.appendText(str);

btnStartStop.setText("Stop");

});

}

**else**

{

stopServer();

Platform.*runLater*(()->{

txtDisplay.appendText("[Server stopped...]\n");

btnStartStop.setText("Start");

});

}

});

Scene scene = **new** Scene(root,400,300);

primaryStage.setTitle("Chatting Server");

primaryStage.setOnCloseRequest(event->stopServer());

primaryStage.setScene(scene);

primaryStage.show();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

<클라이언트>

**package** application;

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**import** java.util.concurrent.\*;

**import** javafx.application.\*;

**import** javafx.event.\*;

**import** javafx.geometry.\*;

**import** javafx.scene.\*;

**import** javafx.scene.control.\*;

**import** javafx.scene.input.\*;

**import** javafx.scene.layout.\*;

**import** javafx.scene.paint.\*;

**import** javafx.scene.text.\*;

**import** javafx.stage.\*;

**public** **class** Ex3\_1 **extends** Application {

Socket socket;

TextArea txtDisplay;

**public** **void** startClient(String ipAddress, **int** port)

{

Thread thread = **new** Thread()

{

**public** **void** run()

{

**try**

{

socket = **new** Socket();

socket.connect(**new** InetSocketAddress(ipAddress, port));

receive();

}

**catch**(Exception e)

{

**if**(!socket.isClosed()) stopClient();

System.***out***.println("[Server connection failure!!]");

Platform.*exit*();

}

}

};

thread.start();

}

**public** **void** stopClient()

{

**try**

{

**if**(socket != **null** && !socket.isClosed())

{

socket.close();

}

}

**catch** (Exception e) { }

}

**public** **void** receive()

{

**while**(**true**)

{

**try** {

InputStream in = socket.getInputStream();

**byte**[] buf = **new** **byte**[100];

**int** len = in.read(buf);

**if**(len == -1) **throw** **new** IOException();

String msg = **new** String(buf, 0, len, "UTF-8");

Platform.*runLater*(()->{txtDisplay.appendText(msg);});

}

**catch** (IOException e) {

stopClient(); **break**;

}

}

}

**public** **void** send(String msg)

{

Thread thread = **new** Thread()

{

**public** **void** run()

{

**try**

{

OutputStream out = socket.getOutputStream();

**byte**[] buf = msg.getBytes("UTF-8");

out.write(buf);

out.flush();

}

**catch** (Exception e)

{

stopClient();

}

}

};

thread.start();

}

**public** **void** start(Stage primaryStage) **throws** Exception

{

BorderPane root = **new** BorderPane();

root.setPadding(**new** Insets(5));

HBox hbox = **new** HBox();

hbox.setSpacing(5);

TextField txtName = **new** TextField();

txtName.setPrefWidth(150);

txtName.setPromptText("Input your nickname.");

TextField txtIP = **new** TextField("localhost");

HBox.*setHgrow*(txtIP, Priority.***ALWAYS***);

TextField txtPort = **new** TextField("4000");

txtPort.setPrefWidth(80);

hbox.getChildren().addAll(txtName, txtIP, txtPort);

root.setTop(hbox);

txtDisplay = **new** TextArea();

txtDisplay.setEditable(**false**);

root.setCenter(txtDisplay);

BorderPane pane = **new** BorderPane();

TextField txtInput = **new** TextField();

txtInput.setPrefWidth(Double.***MAX\_VALUE***);

txtInput.setDisable(**true**);

txtInput.setOnAction(event->{

send(txtName.getText() + "> " + txtInput.getText() + "\n");

txtInput.setText("");

txtInput.requestFocus();

});

Button btnSend = **new** Button("Send");

btnSend.setDisable(**true**);

btnSend.setOnAction(event->{

send(txtName.getText() + "> " + txtInput.getText() + "\n");

txtInput.setText("");

txtInput.requestFocus();

});

Button btnConn = **new** Button("Start");

btnConn.setOnAction(event->{

**if**(btnConn.getText().equals("Start"))

{

**int** port = 4000;

**try**

{

port = Integer.*parseInt*(txtPort.getText());

}

**catch** (Exception e) { }

startClient(txtIP.getText(), port);

Platform.*runLater*(()->{

txtDisplay.appendText("Chatting Started...\n");

});

btnConn.setText("Stop");

txtInput.setDisable(**false**);

btnSend.setDisable(**false**);

txtInput.requestFocus();

}

**else**

{

stopClient();

Platform.*runLater*(()->{

txtDisplay.appendText("Chatting Stopped...\n");

});

btnConn.setText("Start");

txtInput.setDisable(**true**);

btnSend.setDisable(**true**);

}

});

pane.setLeft(btnConn);

pane.setCenter(txtInput);

pane.setRight(btnSend);

root.setBottom(pane);

Scene scene = **new** Scene(root,500,300);

primaryStage.setTitle("Chatting Client");

primaryStage.setScene(scene);

primaryStage.setOnCloseRequest(event->stopClient());

primaryStage.show();

btnConn.requestFocus();

}

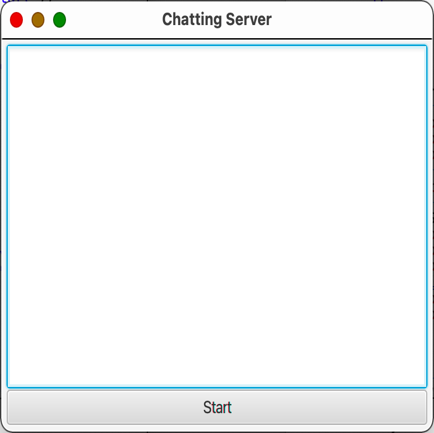
**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

실행 화면 :

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

[ 프로젝트 소감 ]

* Thread Ex 1번의 경우, 지난 학기 자바 프로그래밍 과목에서 기말 프로젝트로 제출한 파리 잡기 게임과 비슷한 느낌으로 코드가 진행되어 손쉽게 코드를 이해하며 구현을 진행할 수 있었습니다.

다만, 그 때와 다르게 JavaFX를 이용하여 구현하였기 때문에 더 다양한 기능이나 쉬워졌다고 느껴지는 기능을 이용할 수 있어 좋았습니다.

* Thread Ex 2번의 경우, AudioClip의 코드를 구현하는데 있어서 애를 먹었습니다.

AudioClip에 해당하는 요소를 java.net 패키지의 URL을 이용하여 긁어온 뒤, java.applet 패키지의 AudioClip을 이용하여 오디오 객체로 만드는 코드였는데, 강의 자료에서 이용한 방법을 그대로 사용하니 제가 전에 배운걸 까먹은건지, 아니면 java가 문제인건지, 사용중인 java버전에서는 반려된다(deprecated)된다는 문구와 함께, 아예 사용 자체가 안되는 문제가 발생했습니다. 이에 꽤 오랜 시간 구글링을 한 끝에, AudioClip 객체를 생성하는 방식을 Applet.newAudioClip(오디오요소); 로 바꾸어 해당 오디오요소가 잘 출력되도록 하였습니다. 해당 사용방식이 제가 사용중인 java버전에서는 곧 사라질 예정이라는 경고메시지만 주고, 사라졌다는 말은 어디에도 없음에도 아예 실행조차 안되어 애를 먹었지만, 구글링을 통해 새로운 정보를 알게 되었으므로 하나 배워간다고 생각했습니다.

* Thread Ex 3번의 경우, 텍스트를 통해 채팅을 할 수 있는 프로그램을 구현하였습니다. 채팅 서버를 구동시켜야만 채팅 클라이언트를 이용할 수 있는 구조였습니다. 상당히 흥미로운 프로그램이었지만, 한 가지 아쉬웠던 점은 한글 채팅까지는 가능하지만 오타를 냈을 경우, 해당 오타가 한글일 경우에는 지워지지 않는 문제가 존재했다는 것이었습니다. 이는 java와 한글간의 호환성 문제로 보고, 추후 더 나은 방법으로 개선시킬 수 있을 것 같았습니다.
* 비록 난이도가 있는 내용이고 프로그램 자체적으로 에러도 많이 나서 마음 고생이 개인적으로 심했으나, 막상 마지막으로 Thread Programming을 통해 그럴듯한 결과물을 제작해보니 나름대로 얻어간 것이 있는 것 같아 뿌듯함을 느꼈습니다. 한 학기 동안 고급 자바 프로그래밍을 강의하시느라 고생 많으셨습니다!