REPORT

동국대학교
dongguk university

- 과목명 : 객체지향프로그래밍
- 담당교수 : 엄진영 교수님
- 학과 : 컴퓨터공학과
- 학번 : 2019112130
- 이름 : 조양진

객체지향프로그래밍 9주차 실습과제 보고서

2019112130 조양진

문제 1.

아래 테스트 클래스를 실행시키면 오른쪽 그림과 같은 문구가 출력이 된다. 해당 코드를 실행시키기 위한 클래스를 작성하시오.

객체배열을 사용하여 서로 다른 객체를 배열 안에 대입하고 있습니다.
아마 기본이 되는 Animal 클래스를 extend해서 Cat, Dog, Cow 클래스를 만들고 bark라는 함수를 오버로딩하면 될 것 같습니다.

- 코드 (week9_01 – Animal.java, Cat.java, Cow.java, Dog.java, Problem1.java)

```
// 2019112130 조양진
// Problem1.java
package week9;

public class Problem1 {
	public static Animal[] animals;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		animals = new Animal[3];
		animals[0] = new Cat();
		animals[1] = new Dog();
		animals[2] = new Cow();

		for(int i=0; i<animals.length; i++) {
			animals[i].bark();
		}
	}
}
```

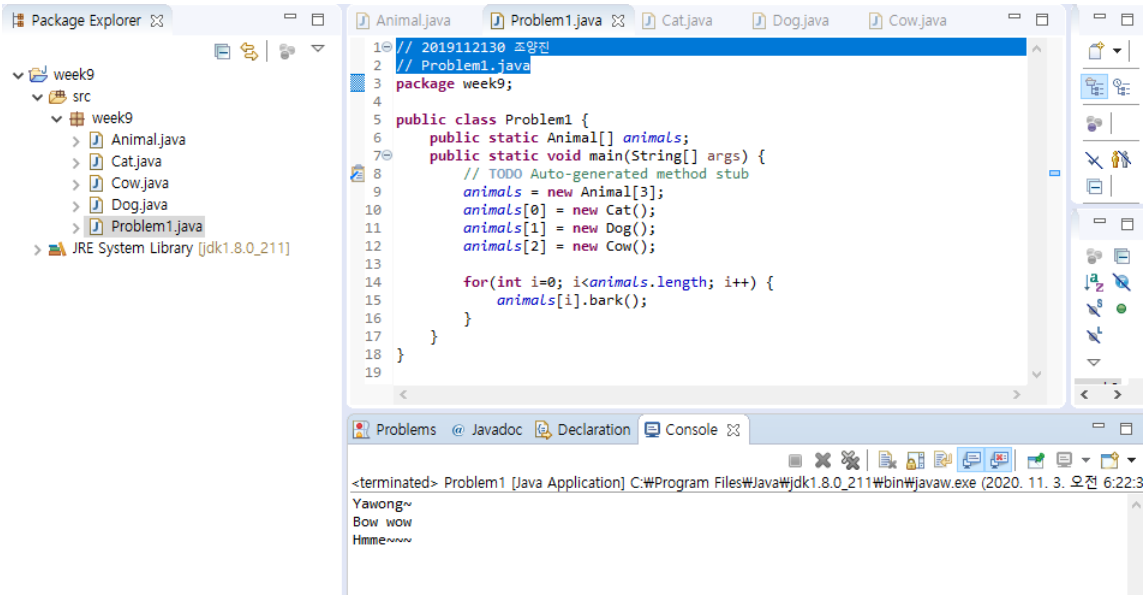
```java
// 2019112130 조양진
// Animal.java
package week9;

public class Animal {
	public void bark() {
		System.out.println("Default Sound");
	}
}
```

```java
// 2019112130 조양진
// Cat.java
package week9;

public class Cat extends Animal {
	public void bark() {
		System.out.println("Yawong~");
	}
}
```

```java
// 2019112130 조양진
// Dog.java
package week9;

public class Dog extends Animal{
	public void bark() {
		System.out.println("Bow wow");
	}
}
```

```java
// 2019112130 조양진
// Cow.java
package week9;

public class Cow extends Animal{
	public void bark() {
		System.out.println("Hmme~~~");
	}
}
```

- 결과 및 분석



문제에서 제시한 그대로 출력됨을 확인할 수 있었습니다.
Cat, Dog, Cow 클래스의 경우 Animal 클래스를 확장해서 쓰지 않으면 Animal 객체배열 animals 에 대입을 할 수 없기 때문에 extend가 필수입니다.

문제 2.

1. https://github.com/bethrobson/Head-First-Java/blob/master/chap15/SimpleChatClientA.java
2. https://github.com/bethrobson/Head-First-Java/blob/master/chap15/VerySimpleChatServer.java

위 2개의 코드를 받아 실행시켜서 작동시켜보고 결과 화면 및 코드 분석을 자세하게 진행하여 보고서를 작성하세요.

- 코드 (week9_02 – SimpleChatClientA.java, VerySimpleChatServer.java)

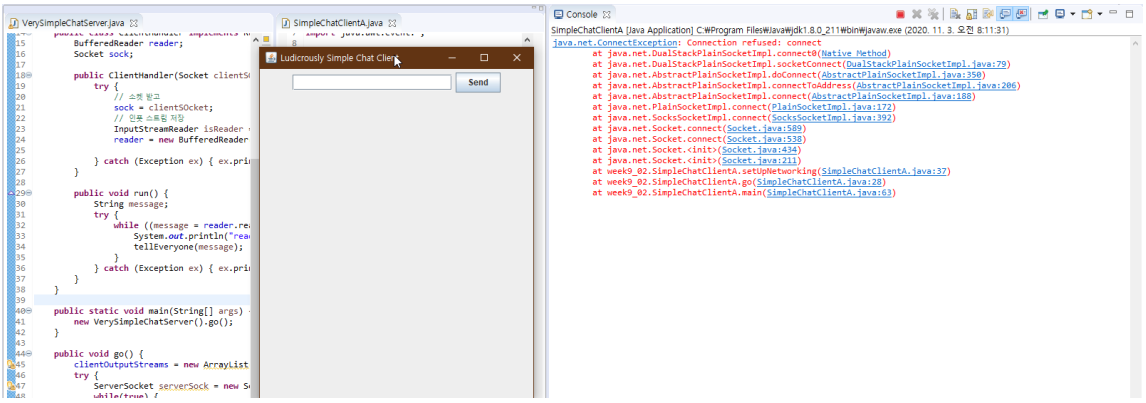
```
// 2019112130 조양진
// VerySimpleChatServer.java
// original code: https://github.com/bethrobson/Head-First-Java/blob/master/chap15/VerySimpleChatServer.java

package week9_02;

import java.io.*;
import java.net.*;
import java.util.*;
```

```
// 채팅 서버 클래스
public class VerySimpleChatServer
{
       // output 스트림용 arraylist
    ArrayList clientOutputStreams;

    // 클라이언트 핸들링
    public class ClientHandler implements Runnable {
       // 버퍼 리더와 소켓 생성
        BufferedReader reader;
        Socket sock;

        public ClientHandler(Socket clientSOcket) {
            try {
             // 소켓 받고
                sock = clientSOcket;
                // 인풋 스트림 저장
                InputStreamReader isReader = new
InputStreamReader(sock.getInputStream());
                reader = new BufferedReader(isReader);

            } catch (Exception ex) { ex.printStackTrace(); }
        }

        public void run() {
            String message;
            try {
             // 메세지 받은거 출력
                while ((message = reader.readLine()) != null) {
                    System.out.println("read " + message);
                    // 출력 함수 호출
                    tellEveryone(message);
                }
            } catch (Exception ex) { ex.printStackTrace(); }
        }
    }
    // 메인 함수
    public static void main(String[] args) {
        new VerySimpleChatServer().go();
    }
    // 서버 시작하면 실행되는 함수
    public void go() {
       // 아웃풋 스트림용 어레이리스트
        clientOutputStreams = new ArrayList();
        try {
             // 클라이언트가 로컬호스트 5000번 포트 쓰니깐 소켓 5000번으로 열었음
            ServerSocket serverSock = new ServerSocket(5000);
            while(true) {
             // 소켓 연결받고
                Socket clientSocket = serverSock.accept();
                PrintWriter writer = new
PrintWriter(clientSocket.getOutputStream());
                clientOutputStreams.add(writer);
```

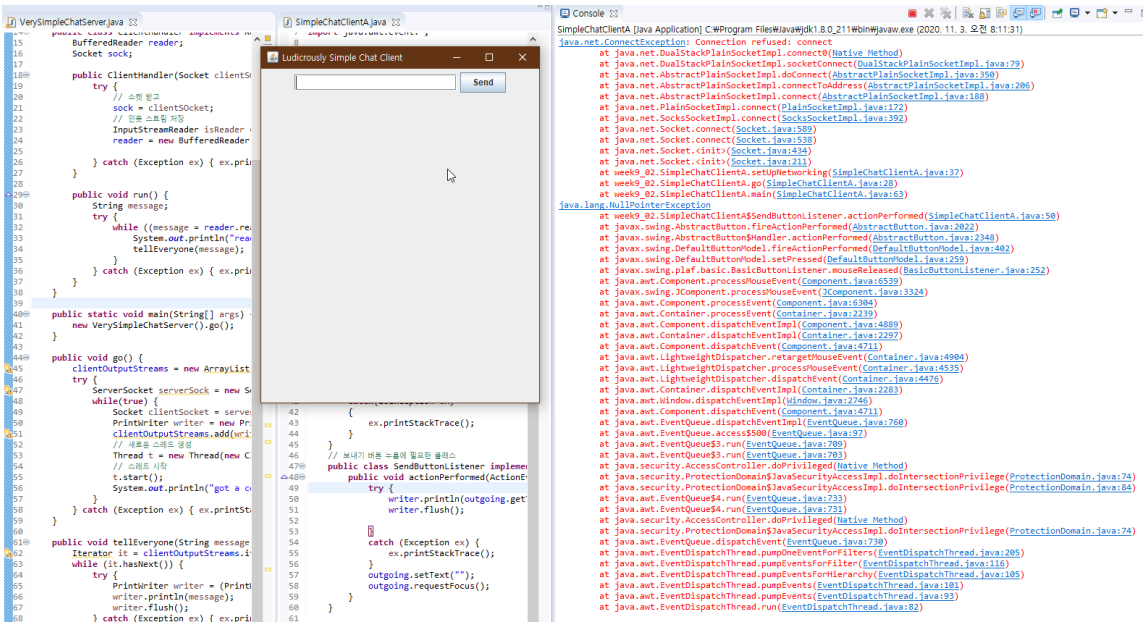
```
                // 새로운 스레드 생성
                Thread t = new Thread(new ClientHandler(clientSocket));
                // 스레드 시작
                t.start();
                System.out.println("got a connection");
            }
        } catch (Exception ex) { ex.printStackTrace(); }
    }
    // 반복해서 입력값이 있으면 출력함
    public void tellEveryone(String message) {
        Iterator it = clientOutputStreams.iterator();
        while (it.hasNext()) {
            try {
                PrintWriter writer = (PrintWriter) it.next();
                writer.println(message);
                writer.flush();
            } catch (Exception ex) { ex.printStackTrace(); }
        }
    }
}



// 2019112130 조양진
// SimpleChatClientA.java
// original code: https://github.com/bethrobson/Head-First-
Java/blob/master/chap15/SimpleChatClientA.java

package week9_02;

import java.io.*;
import java.net.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

// 채팅 클라이언트 A
public class SimpleChatClientA
{
      // 클라이언트 실행시 GUI로 입력하는 곳
    JTextField outgoing;
    PrintWriter writer;
    Socket sock;

    public void go() {
      // GUI 이름 설정
        JFrame frame = new JFrame("Ludicrously Simple Chat Client");
        JPanel mainPanel = new JPanel();
        outgoing = new JTextField(20);
        // 보내기 버튼 생성
        JButton sendButton = new JButton("Send");
        // 버튼에 액션리스너로 SendButtonListener 클래스 연결
        sendButton.addActionListener(new SendButtonListener());
        // 메인 판넬에 텍스트박스 추가
```

```
        mainPanel.add(outgoing);
        // 메인 판넬에 보내기 박스 추가
        mainPanel.add(sendButton);
        frame.getContentPane().add(BorderLayout.CENTER, mainPanel);
        // 네트워크 연결
        setUpNetworking();
        // 사이즈 400*500
        frame.setSize(400, 500);
        frame.setVisible(true);

    }
    // 네트워크 연결
    private void setUpNetworking() {
        try {
             // localhost, 5000포트
            sock = new Socket("127.0.0.1", 5000);
            writer = new PrintWriter(sock.getOutputStream());
            System.out.println("networking established");
        }
        catch(IOException ex)
        {
            ex.printStackTrace();
        }
    }
    // 보내기 버튼 누름에 필요한 클래스
    public class SendButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent ev) {
            try {
                writer.println(outgoing.getText());
                writer.flush();

            }
            catch (Exception ex) {
                ex.printStackTrace();
            }
            outgoing.setText("");
            outgoing.requestFocus();
        }
    }

    public static void main(String[] args) {
        new SimpleChatClientA().go();
    }
}
```

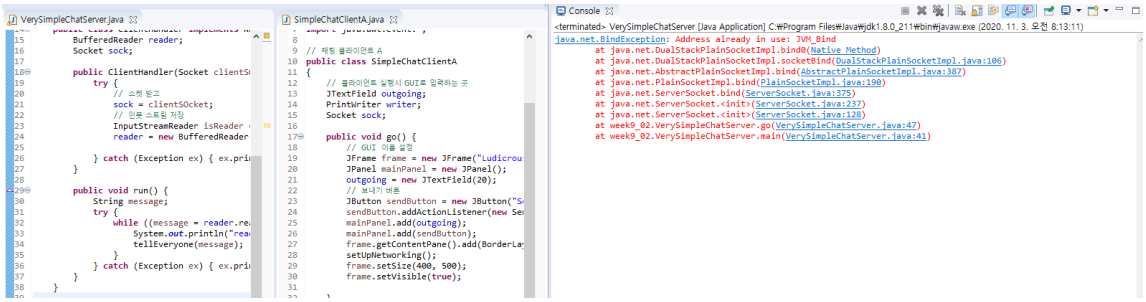
- 결과 및 분석



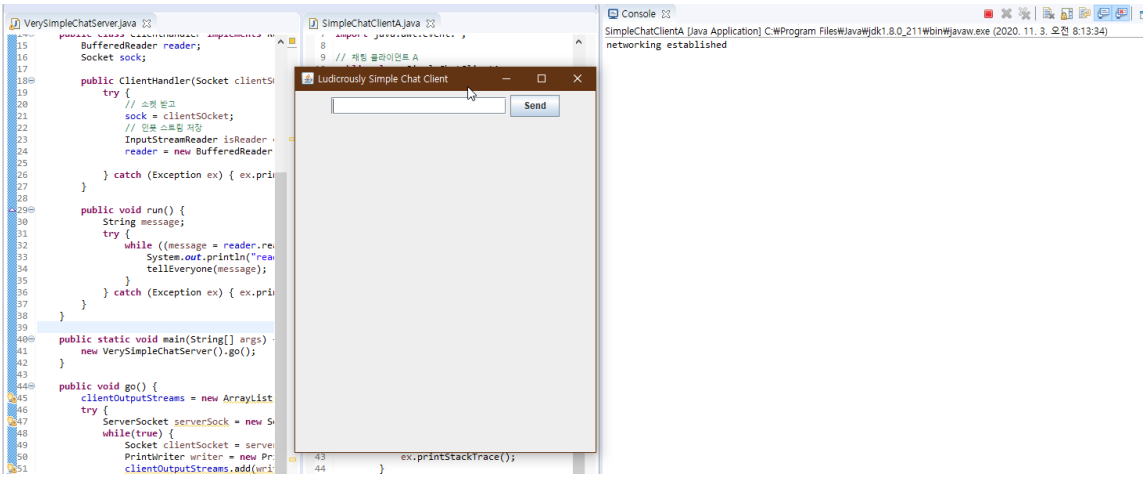
- 서버가 실행되지 않고 클라이언트만 실행 했을 때 서버와 연결되지 못해 오류 발생



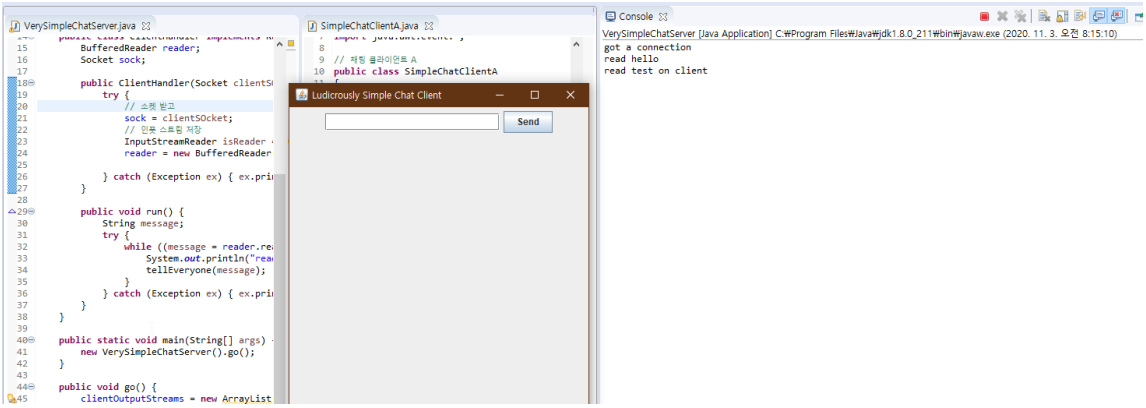
- 당연히 서버가 연결되어 있지 않으니 메시지를 보내도 에러만 뜹니다.



- 서버를 두번 실행했을 경우



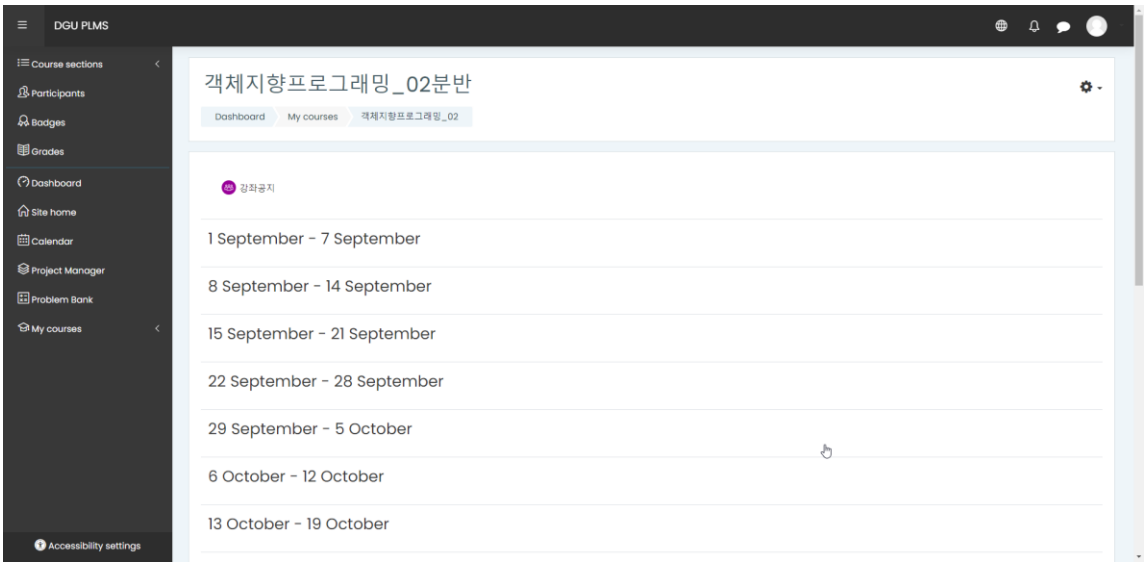
- 서버와 클라이언트가 연결되었을 때



- 클라이언트에 입력하고 Send 버튼을 눌러서 서버에서 스트림을 읽음 – 정상 작동

간단하게 로컬에서 서버를 열고 클라이언트와 연결하여 챗팅을 입력해서 보내고 서버에서 읽는 프로그램이었습니다.

문제 3.



객체지향프로그래밍_02분반에 신청해서 접근할 수 있게되었습니다.