2023년 겨울 자바 사관학교

채팅 서버 / 클라이언트 만들기

목차

- 학습목표
- 1 단계 : 채팅 서버를 구현한다.
  - 분석
  - 설계
  - 구현
- 2 단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.
  - 분석
  - 설계
  - 구현
- 3 단계 : 2 단계에서 작성한 프로그램에 채팅 기능을 구현한다.
  - 분석
  - 설계
  - 구현
- 심화학습

학습목표

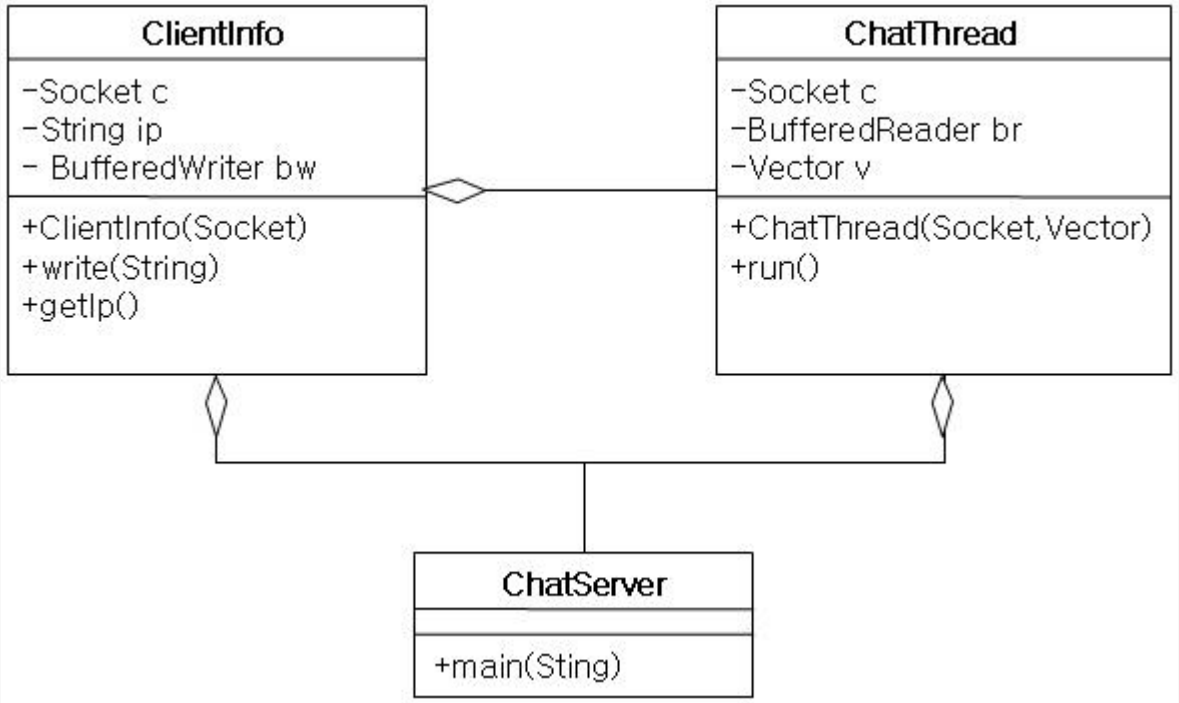
- 자바 Socket 프로그래밍과 쓰레드에 대해 학습한다.
- 익혀야 할 기능
  - Socket 프로그래밍
    - InetAddress 클래스
    - Socket 클래스
    - ServerSocket 클래스
  - 자바 쓰레드

1단계 : 채팅 서버를 구현한다.

- 분석
  - 클라이언트로부터 접속을 받아들일 ServerSocket을 생성하고 새로운 연결 요청이 발생할 때마다 쓰레드를 생성해 처리 함으로서 여러 사용자가 동시에 채팅을 할 수 있도록 한다.
  - 클라이언트 정보 객체의 속성과 동작
    - 속성
      - Socket
      - 클라이언트의 IP
      - 출력 스트림 객체
    - 동작
      - 클라이언트로 문자열 전송
      - IP정보를 반환
  - 채팅 쓰레드 객체의 속성과 동작
    - 속성
      - Socket 입력스트림 객체
      - 클라이언트의 정보를 저장할 벡터 객체
    - 동작
      - 쓰레드 실행을 위한 run() 메서드

1단계 : 채팅 서버를 구현한다.

- 설계
  - 클래스 다이어그램



1단계 : 채팅 서버를 구현한다.

- 중요 메서드의 기능
  - ClientInfo 클래스의 중요 메서드

메서드	기능
public ClientInfo(Socket c)	Socket의 IP를 구해 저장하고 출력스트림을 생성한다.
public boolean write(String str)	인자로 받은 문자열을 출력스트림에 쓴다.
public String getIp()	클라이언트의 IP를 문자열로 반환한다.

  - ChatThread 클래스의 중요 메서드

메서드	기능
public ChatThread(Socket c, Vector v)	객체를 초기화하고 클라이언트로부터 전송된 문자열을 읽기위한 스트림을 생성한다.
public void run()	쓰레드가 실행될 때 호출되는 run() 메서드

1단계 : 채팅 서버를 구현한다.

- 구현
  - ClientInfo.java
    - Socket 프로그래밍을 하기 위해 java.net 패키지를 import 한다.

```
import java.net.*;
```

    - Socket의 getInetAddress() 메서드로 구한 InetAddress 객체의 getHostAddress() 메서드로 클라이언트의 IP 주소를 구해 문자열로 저장한다.
    - Socket의 getOutputStream() 메서드로 출력 스트림을 구하고 이 출력 스트림을 인수로 BufferedWriter 객체를 생성한다.

```
public ClientInfo(Socket c) {
  this.c = c;
  ip = c.getInetAddress().getHostAddress();

  try {
    bw = new BufferedWriter(new OutputStreamWriter(c.getOutputStream()));
  }catch(Exception e) {}
}
```

1단계 : 채팅 서버를 구현한다.

- 인자로 받은 문자열 str 을 BufferedWriter 객체의 write() 메서드의 인자로 사용해 출력 스트림에 문자열을 쓴다.

```
public boolean write(String str) {
  try {
    bw.write(str+"₩n");
    bw.flush();
  }catch(Exception e) {
    return false;
  }
  return true;
}
```

- ChatThread.java
  - 클라이언트로부터 전송된 문자열을 읽기위해 Socket의 getInputStream() 메서드를 사용해 얻은 스트림 객체를 인수로 하여 BufferedReader 객체를 생성한다.

```
try {
  br = new BufferedReader(new InputStreamReader(c.getInputStream()));
}catch(Exception e) {}
```

1단계 : 채팅 서버를 구현한다.

- BufferedReader 객체의 readLine() 메서드를 호출해 입력 스트림 버퍼에 있는 문자열을 읽어온다.

```
try {
  str = br.readLine();
}catch(Exception e) {
  break;
}
```

- 벡터의 객체에 저장되어있는 ClientInfo 객체의 write() 메서드를 호출하여 각 클라이언트에 수신한 문자열을 전송한다.

```
for(int i=0; i<v.size(); i++) {
  ClientInfo ci = (ClientInfo)v.get(i);
  ci.write(str);
}
```

- 출력 스트림에 문자열을 쓸 때 예외가 발생하여 루프를 빠져나오게 되면 클라이언트와의 Socket을 종료하기 위해 close() 메서드를 호출한다.

```
try {
  br.close();
  c.close();
}catch(Exception e) {}
```

1단계 : 채팅 서버를 구현한다.

- ChatServer.java
  - 9000 포트를 사용하는 ServerSocket를 생성한다.

```
ss = new ServerSocket(9000);
```

  - 루프를 돌며 클라이언트의 접속 요청이 생길 때까지 대기를 한다. 클라이언트로부터 접속 요청이 발생하면 서버와 클라이언트의 통신에 사용할 Socket 객체를 생성한다.
  - ClientInfo 객체를 인수로 Runnable 인터페이스를 구현한 ChatThread 객체를 생성한다. 이렇게 생성된 ChatThread 객체를 사용해 Thread 객체를 생성하고 start() 메서드를 호출하여 쓰레드를 실행시킨다.

```
while(true) {
  Socket c = ss.accept();
  …
  ChatThread ct = new ChatThread(c, v);
  Thread t = new Thread(ct);
  t.start();
  …
}
```

1단계 : 채팅 서버를 구현한다.

- ServerSocket에 예외가 발생하면 ServerSocket을 close() 메서드를 사용해 닫고 System.exit()를 호출하여 프로그램을 종료한다.

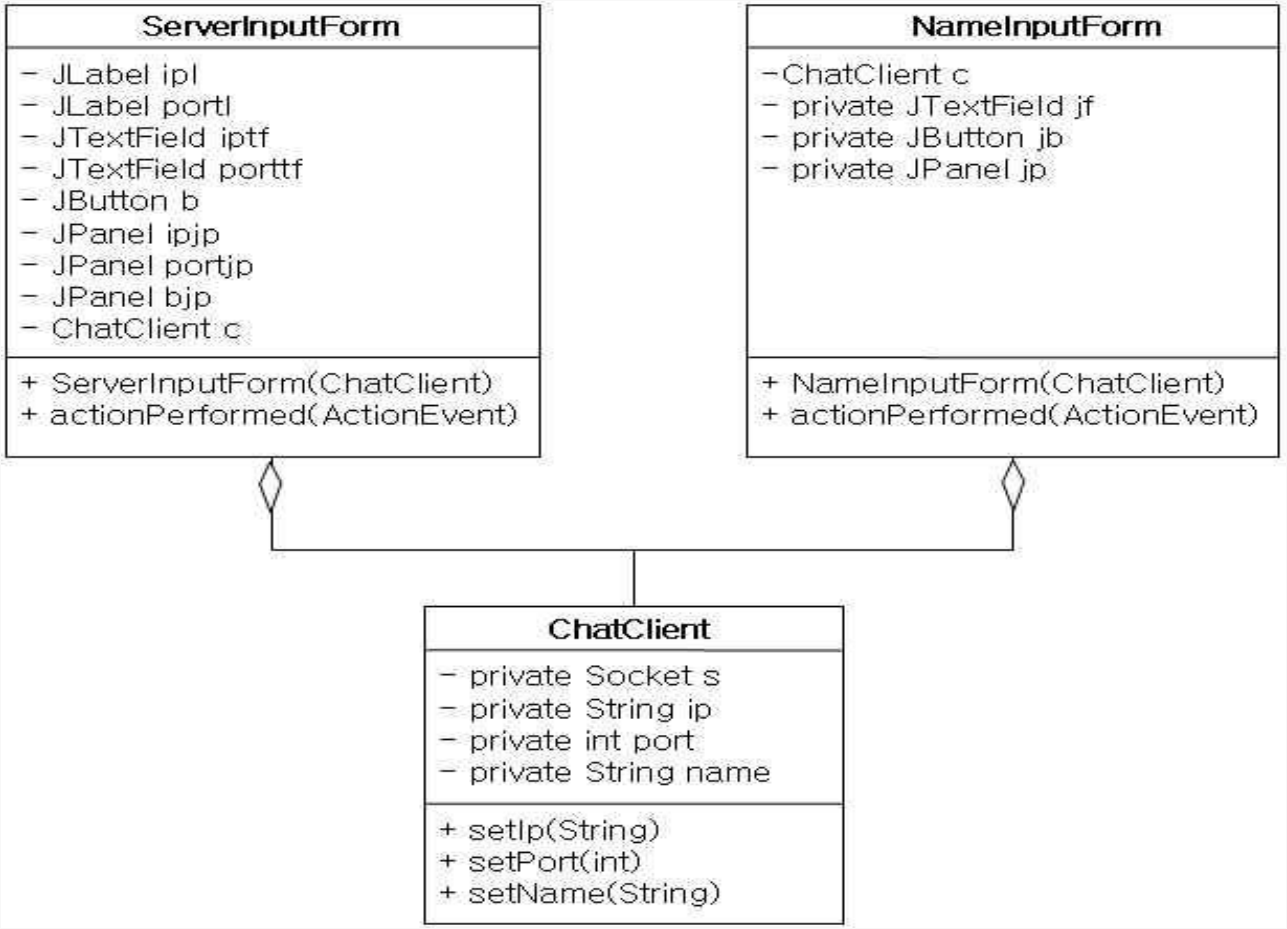
```
catch(Exception e) {
  try {
    ss.close();
    System.exit(0);
  }catch(Exception ioe) {}
}
```

2단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.

- 분석
  - 채팅 서버에 접속하기 위해 IP 정보와 포트 번호를 입력받아 Socket 객체를 생성하고 채팅에 필요한 정보를 초기화 한다.
  - 서버 정보 입력 다이얼로그 객체의 속성과 동작
    - 속성
      - IP 주소 입력을 위한 TextField 객체
      - 포트번호 입력을 위한 TextField 객체
      - 입력된 서버 정보를 저장하기 위한 버튼
    - 동작
      - 입력된 IP 주소와 포트 번호를 저장한다.
  - 이름 입력 다이얼로그 객체의 속성과 동작
    - 속성
      - 대화명 입력을 위한 TextField 객체
      - 입력된 대화명을 저장하기위한 버튼
    - 동작
      - 입력된 대화명을 저장한다.

2단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.

- 설계
  - 클래스 다이어그램



2단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.

- 중요 메서드의 기능
  - ServerInputForm 클래스의 중요 메서드

메서드	기능
public ServerInputForm (ChatClient c)	서버 정보 입력 다이얼로그를 초기화한다.
public void actionPerformed (ActionEvent ae)	버튼 클릭 이벤트가 발생하면 텍스트 필드의 IP 정보와 포트 정보를 저장한다.

  - NameInputForm 클래스의 중요 메서드

메서드	기능
public NameInputForm (ChatClient c)	대화명 입력 다이얼로그를 초기화한다.
public void actionPerformed (ActionEvent ae)	버튼 클릭 이벤트가 발생하면 텍스트 필드의 대화명을 저장한다.

2단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.

- ChatClient 클래스의 중요 메서드

메서드	기능
public void init()	서버 정보 입력창과 대화명 입력창을 호출해 정보를 입력받고 서버와의 통신에 사용할 Socket객체를 생성한다.
public void setIp(String ip)	ServerInputForm에서 호출해 서버의 IP 정보를 저장한다.
public void setPort(int port)	ServerInputForm에서 호출해 서버의 포트 정보를 저장한다.
public void setName(String name)	NameInputForm에서 호출해 사용자의 대화명을 저장한다.
public static void main(String args[])	ChatClient 객체를 생성하고 init() 메서드를 호출하여 채팅 클라이언트를 동작시킨다.

2단계 : 클라이언트의 서버 접속 인터페이스를 구현한다.

- 구현
  - ServerInputForm.java
  - NameInputForm.java
  - ChatClient.java
  - 다이얼로그를 사용해 입력받은 ip 주소와 포트 번호를 인자로 Socket 클래스의 생성자를 호출한다.

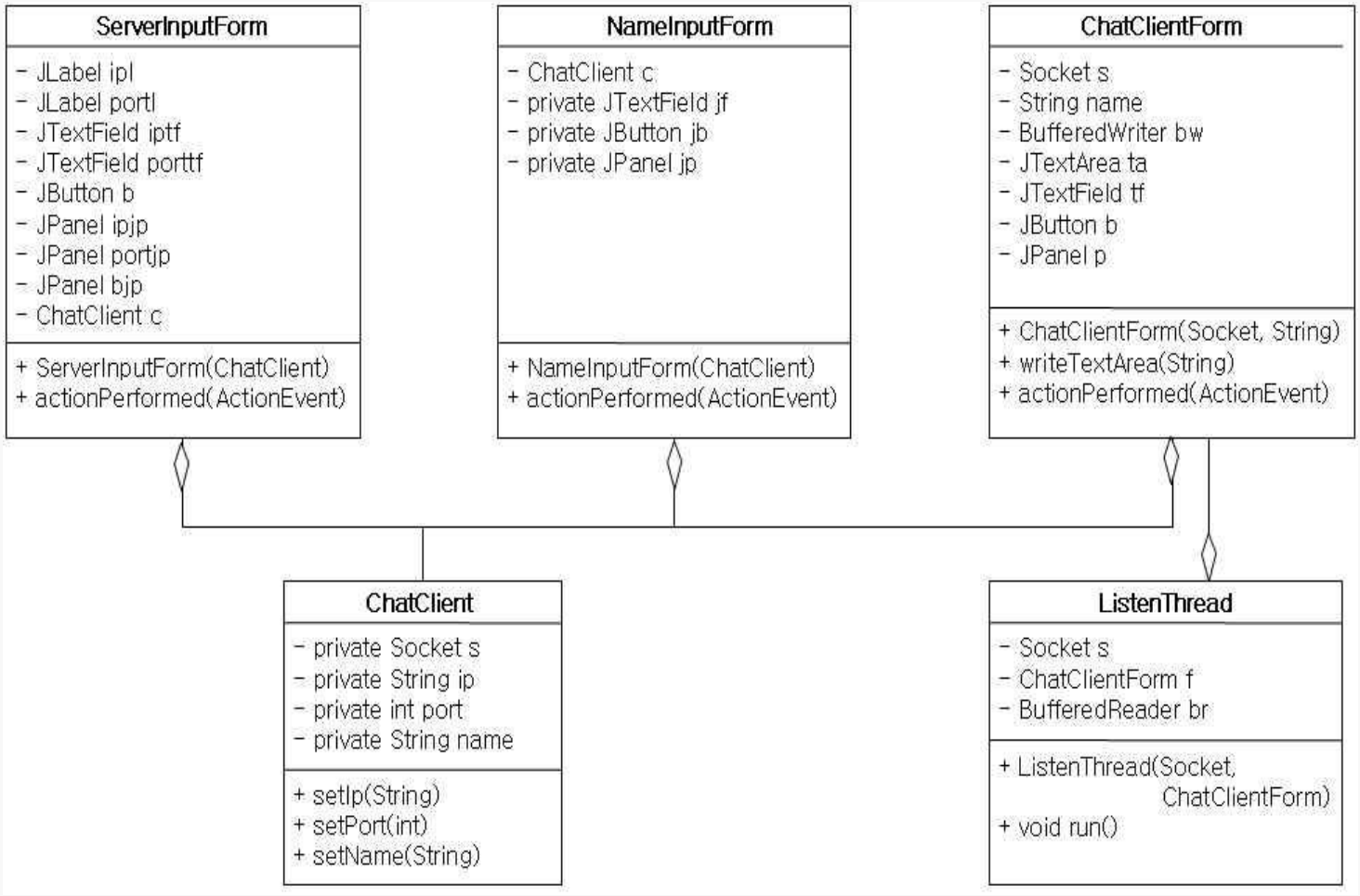
```
try {
  s = new Socket(ip, port);
} catch(Exception e) {}
```

3단계 : 채팅 기능을 구현한다.

- 분석
  - 실제로 문자열을 서버로 전송하고 서버에서는 각 클라이언트로 문자열을 전송하여 채팅 기능을 구현하도록 한다.
  - 채팅 클라이언트 프레임 객체의 속성과 동작
    - 속성
      - 서버와 통신을 위한 Socket 객체
      - 대화명을 저장할 String 객체
      - 문자열을 송신하기 위한 tm트림 객체
      - 채팅 내용을 출력하기위한 TextArea
      - 문자열 입력을 위한 TextField
      - 입력된 문자열을 전송하기 위한 버튼
    - 동작
      - 버튼이 클릭되면 TextField의 내용을 서버에 전송한다.
      - 서버로부터 문자열을 수신하기 위한 쓰레드를 생성한다.
  - ListenThread 객체의 속성과 동작
    - 속성
      - 서버와 통신을 위한 Socket
      - 채팅 클라이언트 객체의 레퍼런스
      - 문자열 수신을 위한 스트림 객체
    - 동작
      - 서버로부터 문자열을 수신하여 TextArea에 출력한다.

3단계 : 채팅 기능을 구현한다.

- 설계
  - 클래스 다이어그램



3단계 : 채팅 기능을 구현한다.

- 중요 메서드의 기능
  - ListenThread 클래스의 중요 메서드

메서드	기능
public ListenThread(Socket s, ChatClientForm f)	BufferedReader 객체를 생성한다.
public void run()	서버로부터 수신된 내용이 있다면 읽어와 ChatClientForm의 TextArea에 쓴다.

  - ChatClientForm 클래스의 중요 메서드

메서드	기능
public ChatClientForm(Socket s, String name)	스위 컴포넌트를 배치한다.
public void writeTextArea (String str)	TextArea에 인수로 받은 문자열 str을 추가한다.
public void actionPerformed (ActionEvent ae)	버튼이 클릭되면 서버에 문자열을 송신한다.

3단계 : 채팅 기능을 구현한다.

- 구현
  - ListenThread.java
    - ListenThread 클래는 Runnable 인터페이스를 구현하고 있기 때문에 반드시 run() 메서드를 구현해 쓰레드가 수행할 작업을 알려주어야 한다.
    - run() 메서드에서는 루프를 돌며 새로운 문자열이 수신되면 BufferedReader 객체를 사용해 문자열을 읽어와 프레임의 JTextArea 객체에 새로운 문자열을 추가한다.

```
public class ListenThread implements Runnable {
  …
  public void run() {
    while(true) {
      try {
        f.writeTextArea(br.readLine());
      }catch(Exception e) {
      }
    }
  }
}
```

3단계 : 채팅 기능을 구현한다.

- Socket 객체의 getInputStream() 메서드를 호출하여 구한 InputStream 객체를 인자로 하여 서버가 전송한 문자열을 읽기 위한 스트림 객체를 생성한다.

```
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
```

- ChatClientForm.java
  - Runnable 인터페이스를 구현한 ListenThread 클래스 객체인 lt를 인자로 새로운 Thread 객체를 생성한다. Thread 객체의 start() 메서드를 호출하면 lt의 run() 메서드가 실행되며 쓰레드로 실행된다.
  - 서버에서 BufferedWriter 객체를 생성한 것과 마차가지로 클라이언트의 Socket에 대해서도 스트림 객체를 생성해야 한다.

```
try {
  ListenThread lt = new ListenThread(s, this);
  Thread t = new Thread(lt);
  t.start();

  bw = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
}catch(Exception e) {}
```

3단계 : 채팅 기능을 구현한다.

- BufferedWtriter 객체의 write(String) 메서드를 호출해서 문자열을 전송하는데 이 때 주의 할 점은 서버에서 수신할 때 사용할 BufferedReader 객체의 readLine() 메서드가 문자열의 끝을 인식할 수 있게 하기 위해 "₩n"를 붙여준다는 것이다

```
public void actionPerformed(ActionEvent ae) {
  try {
    bw.write(name+" : "+tf.getText()+"₩n");
    bw.flush();
    tf.setText("");
  }catch(Exception e) {
    bw.close();
    s.close();
    System.exit(0);
  }
}
```

- ChatClient.java
  - 실제로 채팅 기능을 구현한 ChatClientForm 객체를 생성하고 화면에 출력한다.

```
ChatClientForm ccf = new ChatClientForm(s, name);
ccf.setVisible(true);
ccf.show(true);
```

집중탐구

- Socket 프로그래밍
  - InetAddress 클래스
    - IP 주소를 액세스하는데 필요한 추상층을 제공하는 클래스

메서드	설명
static InetAddress getByName(String host)	호스트의 도메인이나 IP 주소를 인자로 받아 InetAddress 객체를 반환한다.
static InetAddress getLocalHost()	현재 실행되고 있는 로컬 호스트의 InetAddress 객체를 반환한다.
String getHostAddress()	InetAddress의 주소가 멀티캐스트 주소라면 TRUE를 반환하고 아니면 FALSE를 반환한다.
String getHostName()	호스트의 이름을 반환한다.
boolean isMulticastAddress()	호스트의 IP 주소를 문자열로 반환한다.

집중탐구

- Socket 클래스
  - TCP/IP 네트워크에서 프로그래밍을 할 때 Socket API를 이용하게 된다.
  - java.net 패키지에 정의되어 있다.
  - Socket 클래스의 생성자

메서드	설명
Socket(InetAddress address, int port)	address로 설정된 호스트의 port로 연결한다. port는 1-65535 사이의 값으로 입력한다.
Socket(String host, int port)	Socket 객체를 생성하고 인자로 받은 host의 port로 연결한다. host는 도메인 이름이나 IP 주소를 입력한다.
Socket(String host, int port, InetAddress localAddr, int localPort)	Socket 객체를 생성하고, localAddr로 설정된 로컬 주소와 localPort에 바인딩한 뒤, host의 port에 연결한다.

집중탐구

- Socket 클래스
  - Socket 클래스의 중요 메서드

메서드	설명
InputStream getInputStream()	TCP 연결을 통한 스트림 기반의 데이터 송수신에 사용하는 InputStream 객체를 반환한다.
OutputStream getOutputStream()	TCP 연결을 통한 스트림 기반의 데이터 송수신에 사용하는 OutputStream 객체를 반환한다.
void close()	Socket을 닫고 사용되고 있는 모든 리소스를 해제한다. 이 메서드가 호출되기 전에 전송된 모든 데이터는 리모트 호스트에 성공적으로 도달한다.
void shutdownOutput()	통신 중에 출력 스트림을 닫음으로써 Socket을 부분적으로 종료시킨다. 이 기능은 클라이언트가 서버와의 세션을 끊고자 할 때 사용된다.
void shutdownInput()	통신 중에 입력 스트림을 닫음으로써 Socket을 부분적으로 종료시킨다.
InetAddress getInetAddress()	리모트 호스트에 대한 InetAddress 객체를 반환한다.
InetAddress getLocalAddress()	로컬 호스트에 대한 InetAddress를 반환한다.
void setSoTimeOut(int timeout)	읽기 동작이 블록킹되는 상태가 자동으로 취소되기까지의 타임아웃(Time out) 시간을 설정한다.

집중탐구

메서드	설명
int getSoTimeOut()	Socket의 타임아웃 값을 반환한다.
void setReceiveBufferSize(int size)	운영체제에게 수신 버퍼의 크기를 size로 설정하도록 요청한다. 버퍼의 크기를 높이면 큰 데이터를 수신할 때 성능을 높일 수 있지만 운영체제에 따라 무시되는 경우도 있다.
int getReceiveBufferSize()	Socket의 수신 버퍼의 크기를 반환한다.
void setSendBufferSize(int size)	운영체제에게 전송 버퍼의 크기를 size로 설정하도록 요청한다. 버퍼의 크기를 높이면 큰 데이터를 전송할 때 성능을 높일 수 있지만 운영체제에 따라 무시되는 경우도 있다.
int getSendBufferSize()	Socket의 전송 버퍼의 크기를 반환한다.
boolean isClosed()	Socket이 닫혀 있으면 TRUE를 반환하고 연결된 상태라면 FALSE를 반환한다.
boolean isConnected()	Socket이 연결된 상태라면 TRUE를 반환하고 닫혀 있으면 FALSE를 반환한다.

집중탐구

- ServerSocket 클래스
  - 서버가 클라이언트로부터 네트워크를 통한 연결을 받아들일 수 있게 한다.
  - ServerSocket 클래스의 생성자

메서드	설명
ServerSocket(int port)	port로 지정된 로컬 호스트에서 ServerSocket 객체를 생성하고 대기 한다. 동시에 50 개의 연결을 받아들일 수 있다.
ServerSocket(int port, int backlog)	backlog 매개변수를 사용해 우영체제가 큐에 넣을 수 있는 연결의 개수를 정해 줄 수 있다.
Socket(int port, int backlog, InetAddress bindAddr)	bindAddr로 지정된 로컬 주소로 들어온 연결만을 받아들인다. 하나의 호스트에 여러 네트워크 인터페이스가 있을 때 선별적으로 받아들이게 한다.

  - ServerSocket 클래스의 중요 메서드

메서드	설명
Socket accept()	클라이언트가 연결을 요청할 때까지 대기하다가 요청이 생기면 Socket 객체를 반환한다.
void close()	ServerSocket가 더 이상 새로운 연결을 받아들이 않도록 한다.
void setSoTimeOut(int timeout)	accept()를 호출하고 취소할 때까지의 타임아웃 값을 설정한다. 디폴트 값은 0이며 이때는 accept()가 성공할 때까지 계속 대기한다.
int getSoTimeOut()	ServerSocket의 타임아웃 값을 반환한다.

집중탐구

- Thread
  - 멀티쓰레딩(Multi-Threading)
    - 멀티쓰레딩은 하나의 프로그램이 즉, 하나의 프로세스가 동시에 여러 가지 작업을 수행하는 것이다.
    - 멀티쓰레딩으로 프로그램을 작성하게 되면 하나의 프로그램에서 동시에 여러 작업을 수행하는 것이 가능하고 쓰레드간 데이터의 공유가 가능하므로 멀티프로세스보다 효율적으로 프로그래밍 할 수 있다.
    - 또한 새로운 프로세스를 생성할 때 필요한 리소스의 비용을 줄일 수 있는 장점도 가지고 있다

집중탐구

- Thread 클래스
  - 쓰레드를 언어 차원에서 지원
  - 스레드 사용방법에는 java.lang 패키지에 있는 Thread 클래스를 사용하는 방법과 Runnable 인터페이스를 사용하는 방법이 있다
    - 구현 하고자하는 클래스가 Thread 클래스를 상속받도록 한다.
    - run() 메서드를 오버라이딩해 쓰레드로 작동해야 하는 코드를 삽입한다.
    - 클래스의 객체를 생성한다.
    - start() 메서드를 호출하여 쓰레드를 실행시킨다.

```
class 클래스이름 extends Thread {
  …
  public void run() {
    …
  }
  …
}
클래스이름 변수 = new 생성자;
변수.start();
```

    - run() 메서드에서는 while문과 같은 반복문을 사용하는데 그 이유는 run() 메서드가 종료되면 Thread가 자동으로 소멸되기 때문이다.

집중탐구

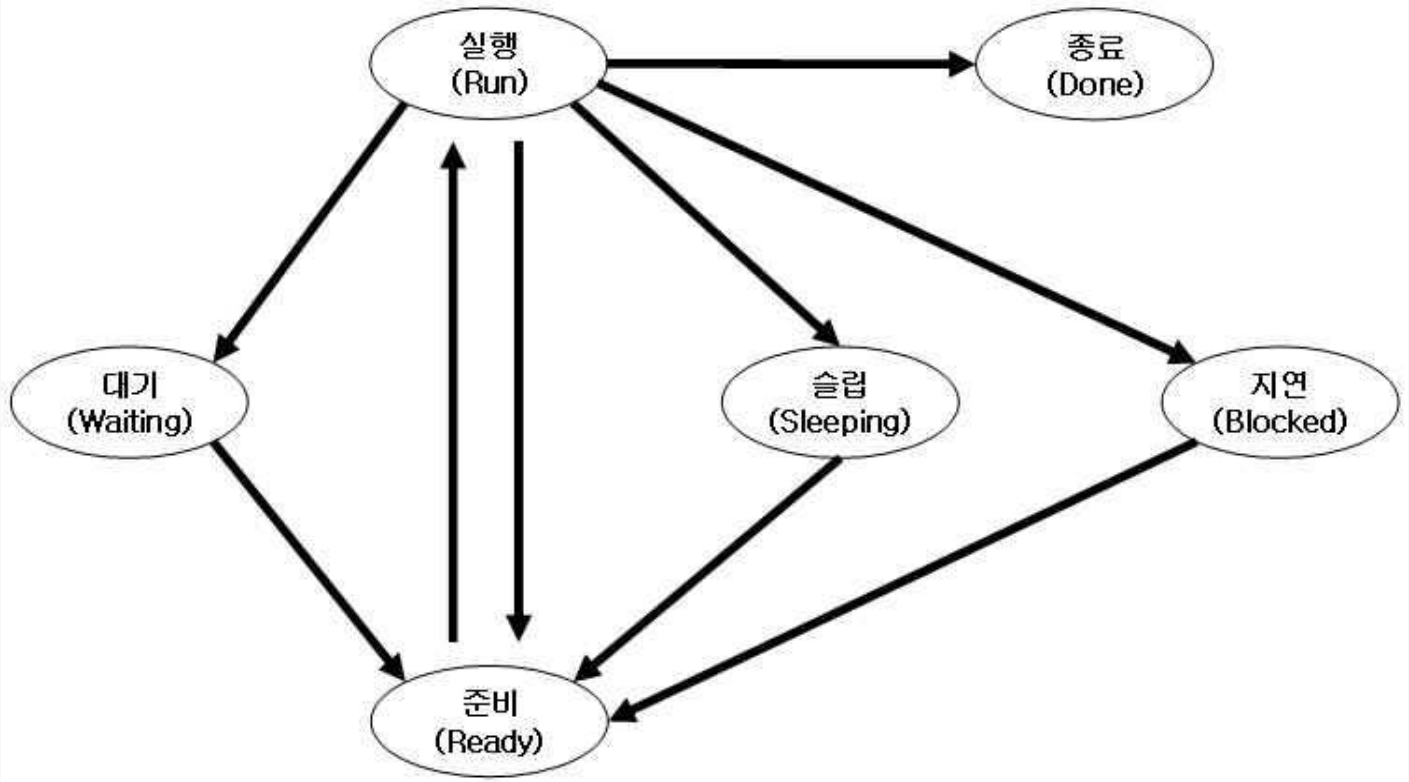
- Runnable 인터페이스
  - 클래스가 이미 다른 클래스를 상속받고 있을때 Runnable 인터페이스를 구현하여 쓰레드를 사용하게 된다
    - 구현 하고자하는 클래스에 Runnable 인터페이스를 구현한다.
    - run() 메서드를 오버라이딩해 쓰레드로 작동해야 하는 코드를 삽입한다.
    - 클래스의 객체를 생성한다.
    - 생성한 객체를 인자로 Thread 객체를 생성한다.
    - 쓰레드 객체의 start() 메서드를 호출하여 쓰레드를 실행시킨다.

```
class 클래스이름 implements Runnable {
  …
  public void run() {
    …
  }
  …
}

클래스이름 변수 = new 생성자;
Thread t = new Thread(변수);
t.start();
```

집중탐구

- 쓰레드의 상태
  - 쓰레드의 상태 천이도



집중탐구

- 준비(Ready)
  - 실행상태로 들어가기 위해 준비하고 있는 상태로 start()메서드가 호출된 쓰레드는 기본적으로 준비상태가 된다.
- 실행(Run)
  - 쓰레드 스케줄러에 의해 선택되면 실행 상태로 들어가게 되며 이때 run() 메서드의 코드가 실행된다.
- 종료(Done)
  - run() 메서드가 종료되어 쓰레드의 실행이 완료된 상태로 한번 종료된 쓰레드는 다시 실행상태가 될 수 없다. 실행상태가 될 수는 없지만 객체가 사라지는 것은 아니므로 객체의 메서드나 멤버변수의 호출은 가능하다.
- 실행정지(Non-Runnable)
  - 실행이 정지된 상태는 다음과 같이 3가지 상태로 구분된다. 이 실행정지 상태는 준비상태와는 다르게 특정 메서드의 호출이나 이벤트가 발생될 때까지 실행되지 못한다.

상태	설명
대기(Waiting)	동기화(Synchronized) 블록 내에서 쓰레드 실행을 중지하고, notify()에 의해 깨워지기를 기다리는 상태
슬립(Sleeping)	CPU의 점유를 중지하고, 특정시간 동안 대기하는 상태
지연(Blocked)	입출력 메서드 등의 호출로 인해 해당 메서드의 종료가 일어날 때까지 대기 하고 있는 상태