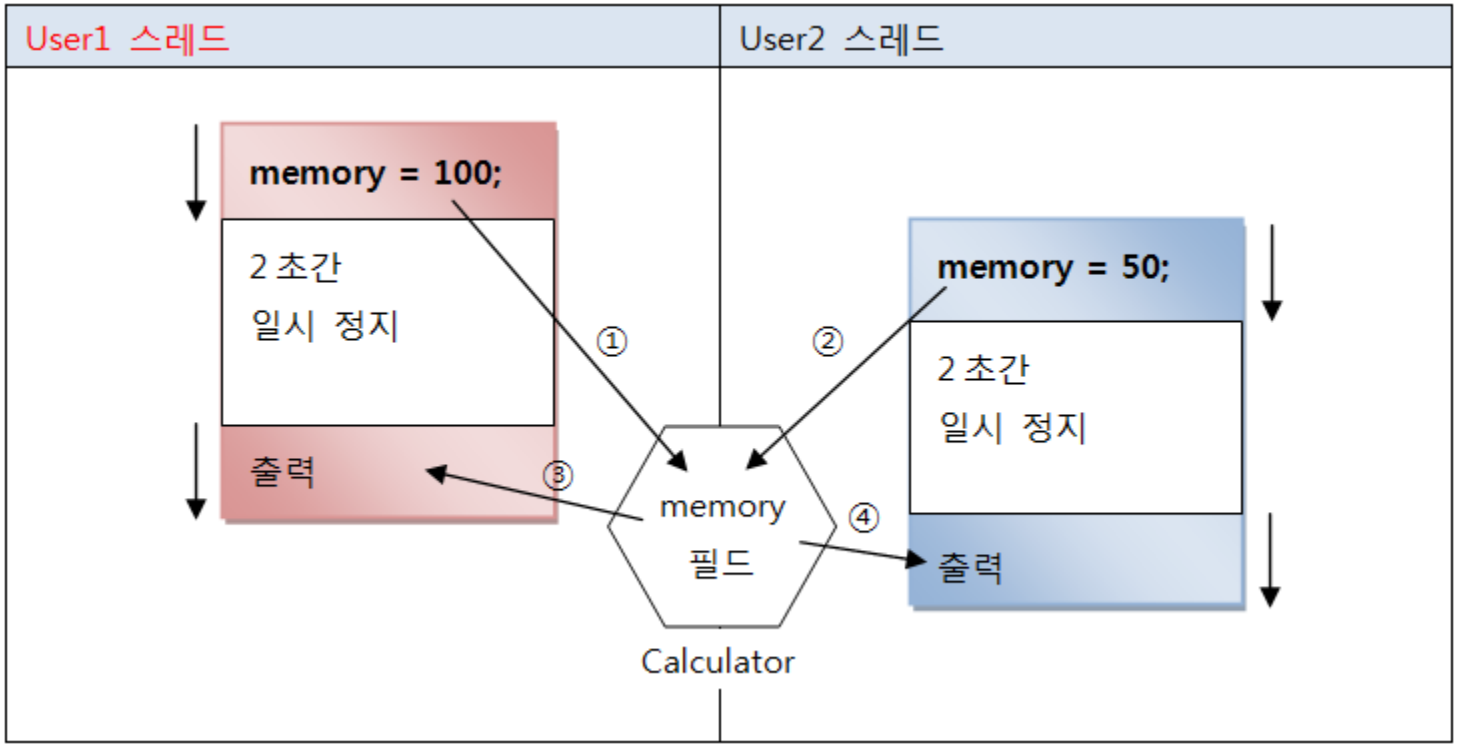
# JAVA언어로 구현하는 스레드/소켓 통신 프로그래밍

# 동기화 메소드와 동기화 블록

- 공유 객체를 사용할 때의 주의할 점
  - 멀티 스레드가 하나의 객체를 공유해서 생기는 오류



# 동기화 메소드와 동기화 블록

- 동기화 메소드 및 동기화 블록 – synchronized
  - 단 하나의 스레드만 실행할 수 있는 메소드 또는 블록
  - 다른 스레드는 메소드나 블록이 실행이 끝날 때까지 대기해야 함
  - 동기화 메소드

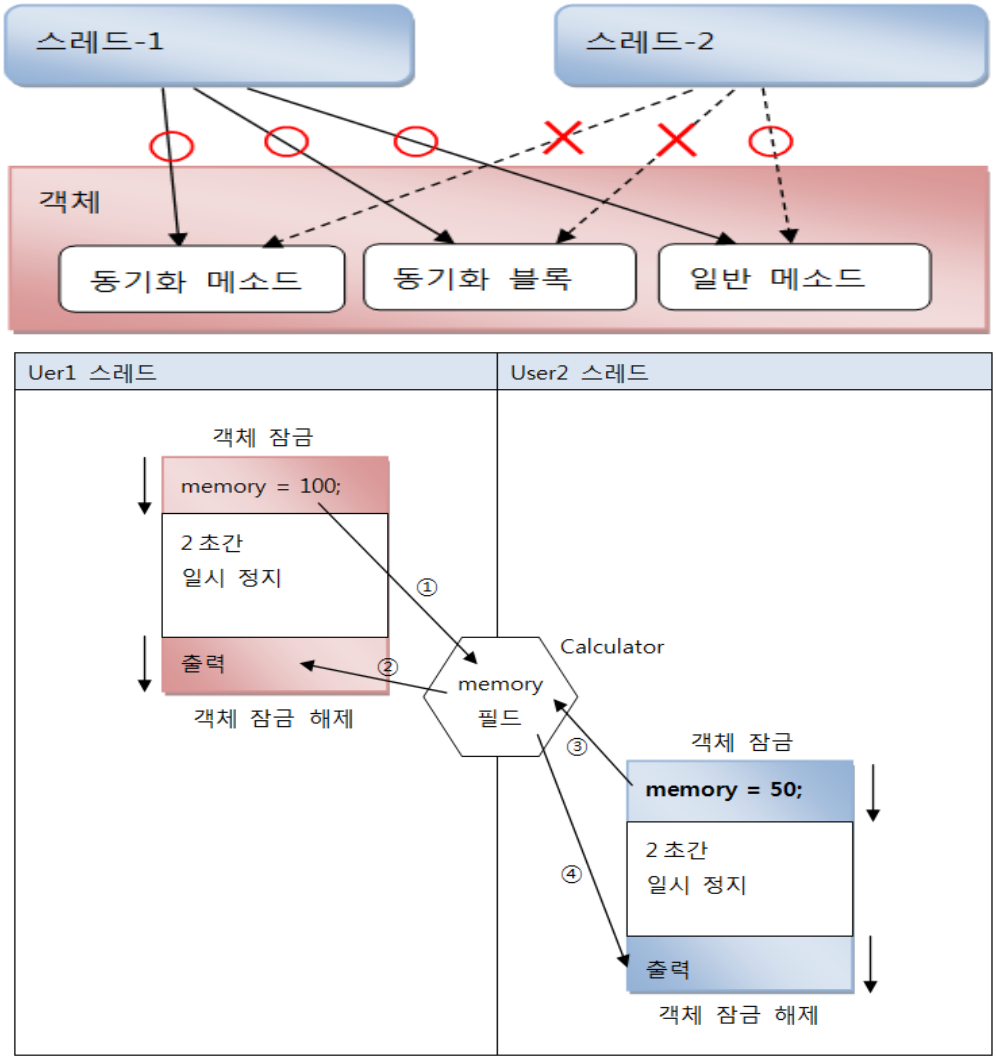
```
public synchronized void method() {
    임계 영역;  //단 하나의 스레드만 실행
}
```

  - 동기화 블록

```
public void method () {
   //여러 스레드가 실행 가능 영역
   ...
   synchronized(공유객체) {
      임계 영역 //단 하나의 스레드만 실행
   }
   //여러 스레드가 실행 가능 영역
   ...
}
```

# 동기화 메소드와 동기화 블록

- 동기화 메소드 및 동기화 블록



# ※ 동기화 예제 2

```
1  package threadExam.Synchronized;
2
3  public class Toilet {
4      public synchronized void openDoor( String name) {
5          System.out.println( name );
6          usingTime();
7          System.out.println( "아~~~~! 시원해" );
8      }
9
10     public void usingTime() {
11     for( int i=0 ; i<100000000 ; i++ ) {
12         if( i == 10000 ) {
13         System.out.println( "끄으응" );
14         }
15         }
16     }
17 }
```

# ※ 동기화 예제 2

```
1  package threadExam.Synchronized;
2
3  public class Family extends Thread {
4      Toilet toilet;
5      String who;
6
7      public Family( String name, Toilet t ) {
8          who = name;
9          toilet = t;
10     }
11
12     public void run() {
13         toilet.openDoor( who);
14     }
15 }
```

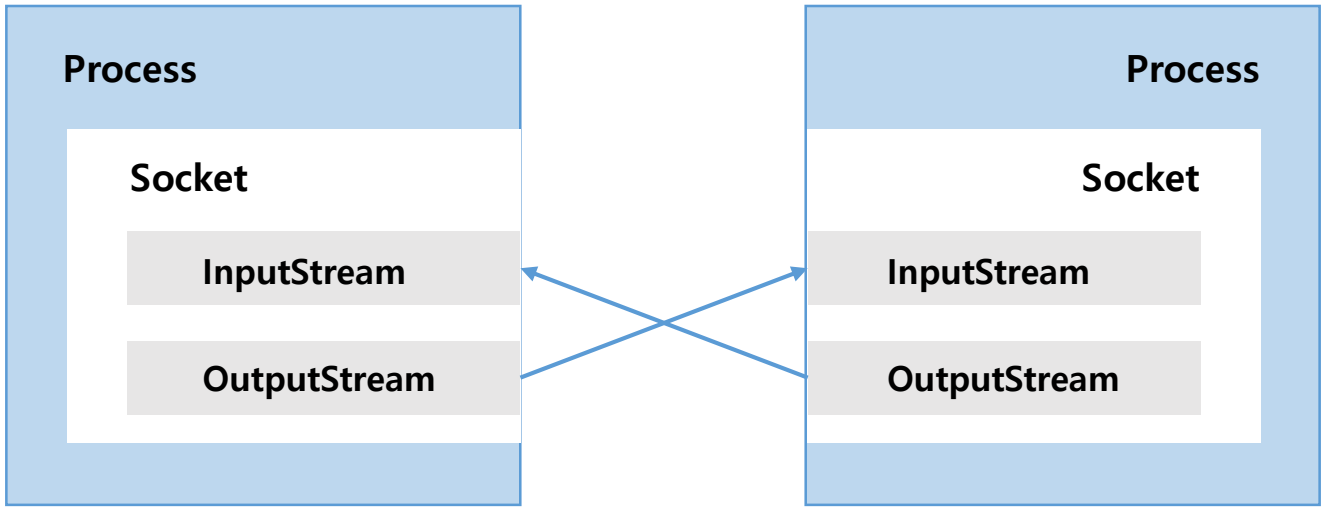
# ※ 동기화 예제 2

```
1  package threadExam.Synchronized;
2
3  public class ManageToilet {
4      public static void main( String[] args ) {
5          Toilet t = new Toilet();
6
7          Family father = new Family("아버지", t );
8          Family mother = new Family("어머니", t );
9          Family sister = new Family("누나", t );
10         Family brother = new Family("형", t );
11         Family me = new Family("나", t );
12
13         father.start();
14         mother.start();
15         sister.start();
16         brother.start();
17         me.start();
18     }
19 }
```

```
<terminated> ManageToilet (1)
아버지
끄으응
아~~~~! 시원해
나
끄으응
아~~~~! 시원해
형
끄으응
아~~~~! 시원해
누나
끄으응
아~~~~! 시원해
어머니
끄으응
아~~~~! 시원해
```

# 자바 소켓 통신

- ServerSocket과 Socket
  - ServerSocket : 소켓간의 연결만 처리
  - Socket: 프로세스간의 실제 통신을 담당,
    InputStream과 OutputStream을 가지고 있음



# 자바 소켓 통신 예제(서버_1)

```
1.  package tcpexam;

2.  import java.io.IOException;
3.  import java.net.InetSocketAddress;
4.  import java.net.ServerSocket;
5.  import java.net.Socket;

6.  public class ServerExample {
7.      public static void main(String[] args) {
8.          ServerSocket serverSocket = null;
9.          try {
10.             //serverSocket = new ServerSocket(5001);       // 생성자에 바인딩 포트를 대입하고 객체를 생성한다.

11.             sererSocket = new ServerSocket();
12.             InetSocketAddress ipep = new InetSocketAddress(5001);
13.             serverSocket.bind(ipep);                      // 디폴트 생성자로 객체를 생성한 후 포트바인딩을 위해 bind()를 호출한다.
14.
15.             while(true) {
16.                 System.out.println( "[연결 기다림]");
17.                 Socket socket = serverSocket.accept();
18.                 InetSocketAddress isa = (InetSocketAddress) socket.getRemoteSocketAddress();
19.                 System.out.println("[연결 수락함] " + isa.getAddress().getHostAddress());
20.             }
21.         } catch(Exception e) {e.printStackTrace();}
22.
23.         if(!serverSocket.isClosed()) {
24.             try {
25.                 serverSocket.close();
26.             } catch (IOException e1) {}
27.         }
28.     }
29. }
```

# 자바 소켓 통신 예제(클라이언트_1)

```
1.  package tcpexam;

2.  import java.io.IOException;
3.  import java.net.InetSocketAddress;
4.  import java.net.Socket;

5.  public class ClientExample {
6.      public static void main(String[] args) {

7.          Socket socket = null;

8.          try {
9.              socket = new Socket();

10.             System.out.println( "[연결 요청]");
11.             socket.connect(new InetSocketAddress("localhost", 5001));
12.             System.out.println( "[연결 성공]");
13.         } catch(Exception e) {e.printStackTrace();}
14.
15.         if(!socket.isClosed()) {
16.             try {
17.                 socket.close();
18.             } catch (IOException e1) {}
19.         }
20.     }
21. }
```

# 자바 소켓 통신 예제(서버_2)

```
1.  package data_write_read;

2.  import java.io.IOException;
3.  import java.io.InputStream;
4.  import java.io.OutputStream;
5.  import java.net.InetSocketAddress;
6.  import java.net.ServerSocket;
7.  import java.net.Socket;

8.  public class ServerExample {
9.      public static void main(String[] args) {
10.         ServerSocket serverSocket = null;
11.         try {
12.             serverSocket = new ServerSocket();
13.             serverSocket.bind(new InetSocketAddress("localhost", 5001));
14.             while(true) {
15.                 System.out.println( "[연결 기다림]");
16.                 Socket socket = serverSocket.accept();
17.                 InetSocketAddress isa
18.                 = (InetSocketAddress) socket.getRemoteSocketAddress();
19.                 System.out.println("[연결 수락함] " + isa.getHostName());
20.
21.                 byte[] bytes = null;
22.                 String message = null;
23.
24.                 InputStream is = socket.getInputStream();
25.                 bytes = new byte[100];
26.                 int readByteCount = is.read(bytes);
27.                 message = new String(bytes, 0, readByteCount, "UTF-8");
28.                 System.out.println("[데이터 받기 성공]: " + message);
29.                 is.close();
30.                 socket.close();
31.             }
32.         } catch(Exception e) {}
33.
34.         if(!serverSocket.isClosed()) {
35.             try {
36.                 serverSocket.close();
37.             } catch (IOException e1) {}
38.         }
39.     }
40. }
```

# 자바 소켓 통신 예제(클라이언트_2)

```
1.  package data_write_read;

2.  import java.io.IOException;
3.  import java.io.InputStream;
4.  import java.io.OutputStream;
5.  import java.net.InetSocketAddress;
6.  import java.net.Socket;

7.  public class ClientExample {
8.      public static void main(String[] args) {
9.          Socket socket = null;
10.         try {
11.             socket = new Socket();
12.             System.out.println( "[연결 요청]");
13.             socket.connect(new InetSocketAddress("localhost", 5001));
14.             System.out.println( "[연결 성공]");
15.
16.             byte[] bytes = null;
17.             String message = null;
18.
19.             OutputStream os = socket.getOutputStream();
20.             message = "Hello Server";
21.             bytes = message.getBytes("UTF-8");
22.             os.write(bytes);
23.             os.flush();
24.             System.out.println( "[데이터 보내기 성공]");
25.             os.close();
26.         } catch(Exception e) {}
27.
28.         if(!socket.isClosed()) {
29.             try {
30.                 socket.close();
31.             } catch (IOException e1) {}
32.         }
33.     }
34. }
```

# 자바 소켓 통신 예제(서버_3)

```
1.  package echoExam;

2.  import java.io.BufferedReader;
3.  import java.io.IOException;
4.  import java.io.InputStream;
5.  import java.io.InputStreamReader;
6.  import java.io.OutputStream;
7.  import java.io.PrintWriter;
8.  import java.net.ServerSocket;
9.  import java.net.Socket;

10. public class EchoServer {

11.     private ServerSocket server;

12.     public EchoServer(int port) throws IOException {
13.         server = new ServerSocket(port);
14.     }

15.     public void service() throws IOException {
16.         System.out.println("EchoServer is ready.");

17.         Socket client = server.accept();

18.         InputStream is = client.getInputStream();
19.         OutputStream os = client.getOutputStream();
20.         BufferedReader in
21.                         = new BufferedReader(new InputStreamReader(is));
22.         PrintWriter out = new PrintWriter(os, true);
23.         while (true) {
24.             String msg = in.readLine();
25.             System.out.println(msg);
26.             if (msg.equals("bye")) {
27.                 break;
28.             }
29.             out.println(">>" + msg);
30.         }
31.     }
32.
33.     public void close() throws IOException{
34.         server.close();
35.     }
36.
37.     public static void main(String[] args) {
38.         try{
39.             EchoServer es = new EchoServer(1289);
40.             es.service();
41.             es.close();
42.         }catch(Exception e){
43.             e.printStackTrace();
44.         }
45.     }
46. }
```

# 자바 소켓 통신 예제(클라이언트_3)

```
1.  package echoExam;

2.  import java.io.BufferedReader;
3.  import java.io.IOException;
4.  import java.io.InputStream;
5.  import java.io.InputStreamReader;
6.  import java.io.OutputStream;
7.  import java.io.PrintWriter;
8.  import java.net.Socket;

9.  public class EchoClient {
10.     private Socket socket;

11.     public EchoClient(String host, int port) throws Exception {
12.         socket = new Socket(host, port);
13.     }

14.     public void echo() throws IOException {
15.         OutputStream os = socket.getOutputStream();
16.         InputStream is = socket.getInputStream();
17.         BufferedReader in
18.             = new BufferedReader(new InputStreamReader(is));
19.         PrintWriter out = new PrintWriter(os, true);
20.         BufferedReader con
21.             = new BufferedReader(   new InputStreamReader(System.in));
22.         while (true) {
23.             String msg = con.readLine();
24.             out.println(msg);
25.             if (msg.equals("bye")) {
26.                 break;
27.             }
28.             System.out.println(in.readLine());  // 서버로부터 받은 내용을 화면에 출력합니다.
29.         }
30.     }

31.     public void close() throws IOException{
32.         socket.close();
33.     }
34.
35.     public static void main(String[] args) {
36.         try{
37.             EchoClient ec;
38.             System.out.println("메시지를 입력하세요");
39.             ec = new EchoClient("localhost", 1289);
40.             ec.echo();
41.             ec.close();
42.         }catch(Exception e){
43.             e.printStackTrace();
44.         }
45.     }
46. }
```