Socket Programming(in Java)

TA 蕭博文

@ Intro to Computer Network 2016.3.24

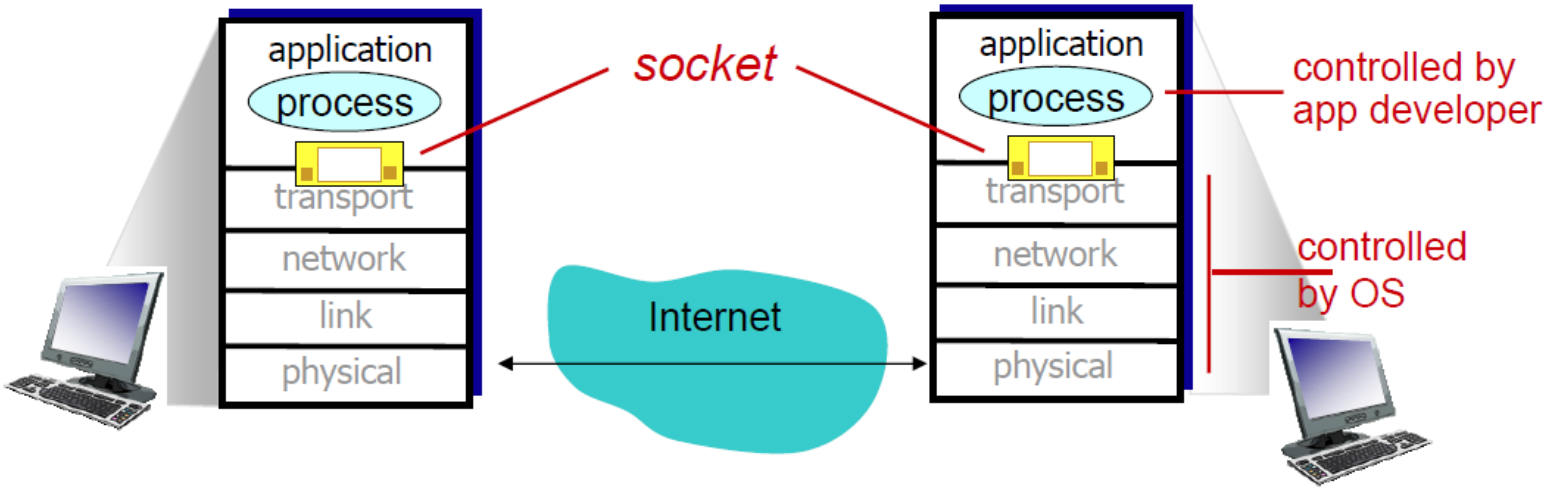
Outline

- Socket Concept
- Quick Java Overview
- TCP Socket Example
- UDP Socket Example

Socket Concept (1/2)

goal: learn how to build client/server applications that communicate using sockets

socket: door between application process and end-end-transport protocol



Socket Concept (2/2)

Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

Application Example:

1. Client reads a line of characters (data) from its keyboard and sends the data to the server.
2. The server receives the data and converts characters to uppercase.
3. The server sends the modified data to the client.
4. The client receives the modified data and displays the line on its screen.

Java: HelloWorld

- Development Environment Setting: Please refer to
  - Eclipse(https://www.eclipse.org/downloads/)
  - JDK8(https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_Howto.html)

- A HelloWorld Program

```
1  // HelloProg.java
2
3  import java.io.*;
4
5  class HelloProg {
6     public static void main(String args[]) throws Exception {
7        System.out.println("Hello world!");
8     }
9  }
```

Java: Class (1/2)

```
 5  class Professor {
 6      // constructor
 7      public Professor(String name) {
 8          this.name = name;
 9      }
10      // instance methods
11      public void setRank(int r) {
12          rank = r;
13      }
14      public void print() {
15          System.out.println("Prof " + name + ": Rank " + rank);
16      }
17      String name;
18      int rank;
19
20      // class methods
21      public static String GetTitle() {
22          return Title;
23      }
24      static String Title = "Ph.D";
25  }
```

Java: Class (2/2)

```
1  // Ex_Class.java
2  import java.io.*;
3
```

⋮ (class Professor Definition)

```
26 class Ex_Class {
27    public static void main(String args[]) throws Exception {
28       String Prof_title = Professor.GetTitle();
29       System.out.println("A professor usually has " + Prof_title + " title.");
30
31       Professor prof = new Professor("Liao");
32       prof.setRank(5);
33       prof.print();
34    }
35 }
```

Java: Exception Handling

```java
// Ex_Exception.java
import java.io.*;

class Professor {
   public Professor(String name) {
      this.name = name;
   }
   public void setRank(int r) throws Exception {
      if (r > 5 || r < 0)
         throw new IllegalArgumentException("Rank should between 0~5 !");
      rank = r;
   }
   public void print() {
      System.out.println("Prof " + name + ": Rank " + rank);
   }
   String name;
   int rank;
}

class Ex_Exception {
   public static void main(String args[]) {
      try {
         Professor liao = new Professor("WJ Liao");
         liao.setRank(100);
         liao.print();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Java: Other Resources

- Syntax Difference Cheat Sheet (vs. C++)
  http://www.cprogramming.com/tutorial/java/syntax-differences-java-c++.html

- API reference
  https://docs.oracle.com/javase/8/docs/api/

Socket programming

Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

Application Example:

1. Client reads a line of characters (data) from its keyboard and sends the data to the server.
2. The server receives the data and converts characters to uppercase.
3. The server sends the modified data to the client.
4. The client receives the modified data and displays the line on its screen.

Socket programming *with TCP*

client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

client contacts server by:

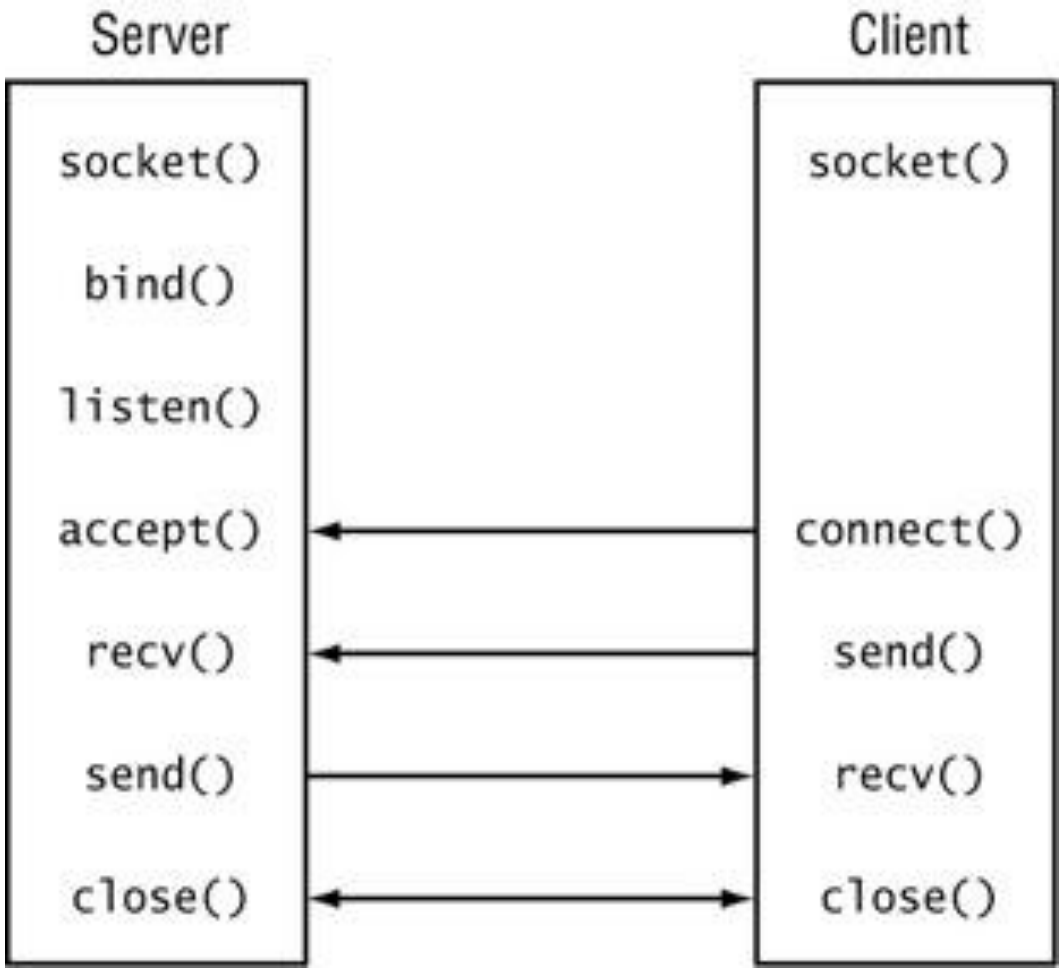
- Creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients (more in Chap 3)

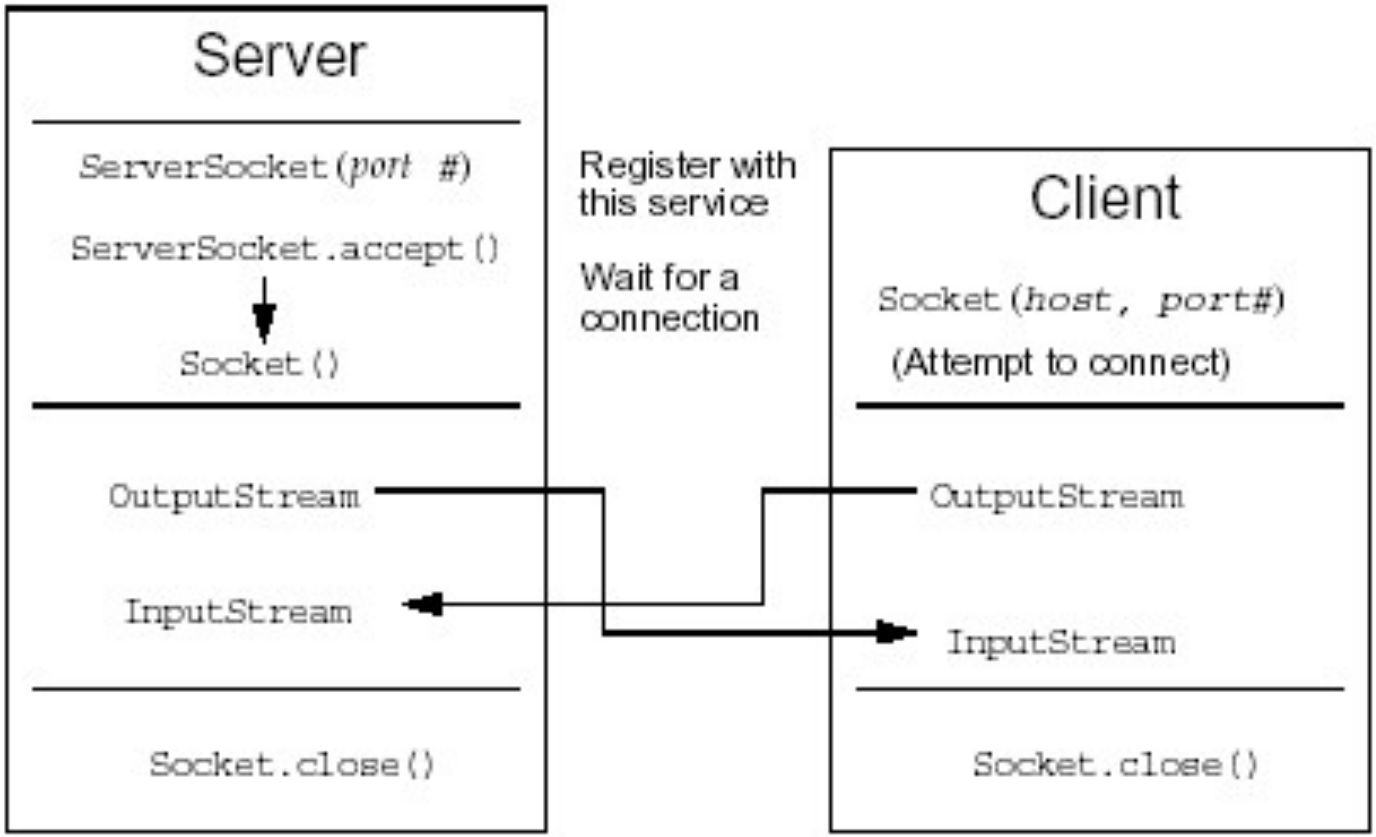
application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

General Flow Chart (TCP)



Flow Chart (TCP) in Java



```java
// TCPClient.java
import java.io.*;
import java.net.*;

class TCPClient {
    public static void main(String args[]) throws Exception {
        Socket clientSocket = new Socket("127.0.0.1", 9090);

        // Read a sentence from User input
        System.out.print("INPUT: ");
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        String sentence = inFromUser.readLine();

        // Write the sentence to Server
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
        outToServer.writeBytes(sentence + '\n');

        // Read the modified sentence from Server
        BufferedReader inFromServer =
            new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        String modifiedSentence = inFromServer.readLine();

        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close();
    }
}
```

```java
// TCPServer.java
import java.io.*;
import java.net.*;

class TCPServer {
    public static void main (String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(9090);
        System.out.println("Server Ready..");
        while (true) {
            Socket socket = welcomeSocket.accept();
            System.out.println("A Client Connect.");

            // Read a sentence from Client
            BufferedReader inFromClient =
                new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String sentence = inFromClient.readLine();
            System.out.println("RECV: " + sentence);

            // Captalize the received sentence
            String modifiedSentence = sentence.toUpperCase() + '\n';
            System.out.println("MODIFY TO: " + modifiedSentence);

            // Write the modified sentence to Client
            DataOutputStream outToClient =
                new DataOutputStream(socket.getOutputStream());
            outToClient.writeBytes(modifiedSentence);
        }
    }
}
```

Socket programming *with UDP*

UDP: no "connection" between client & server

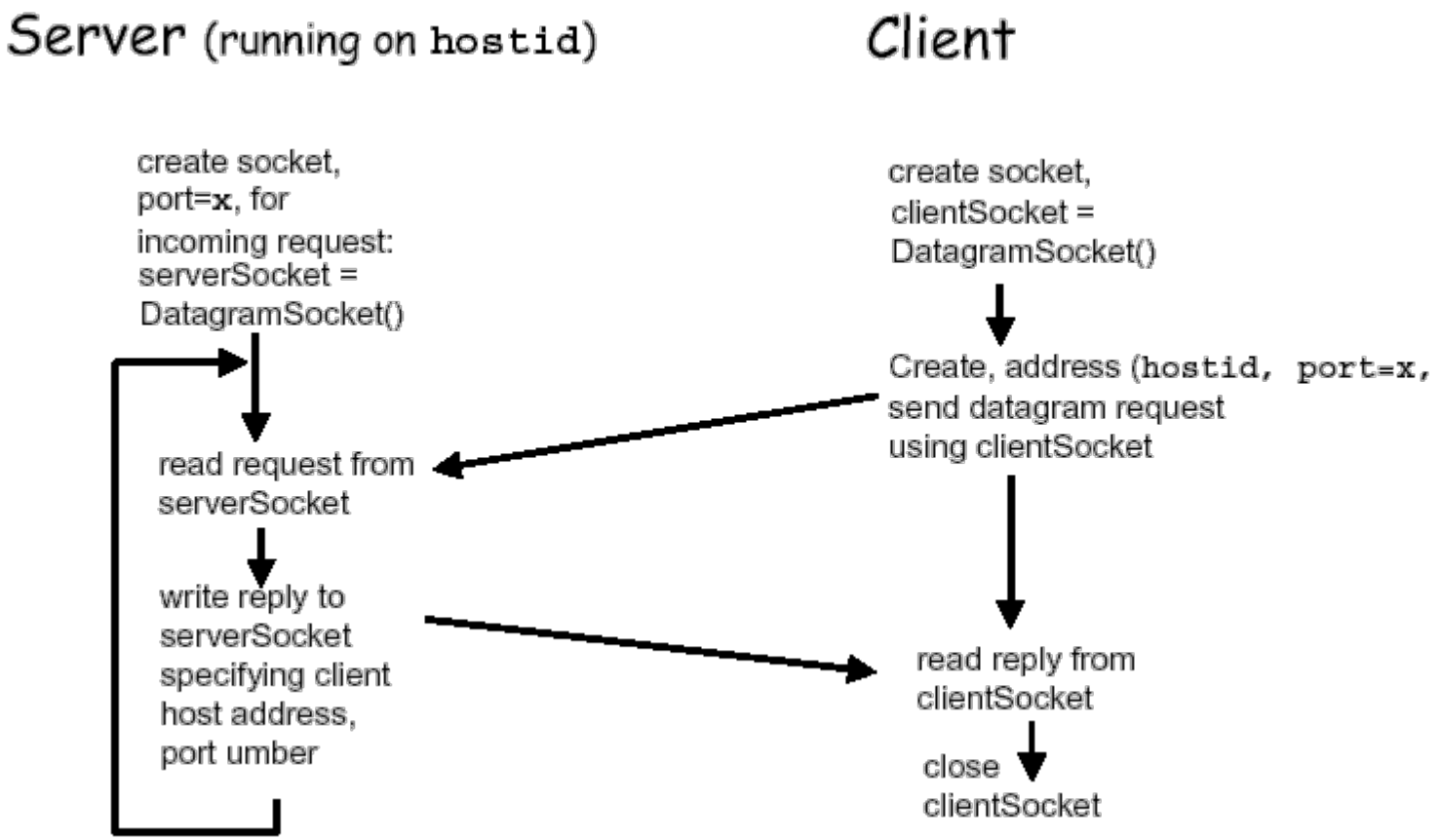
- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- rcvr extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- UDP provides *unreliable* transfer of groups of bytes ("datagrams") between client and server

UDP Socket (cont'd)

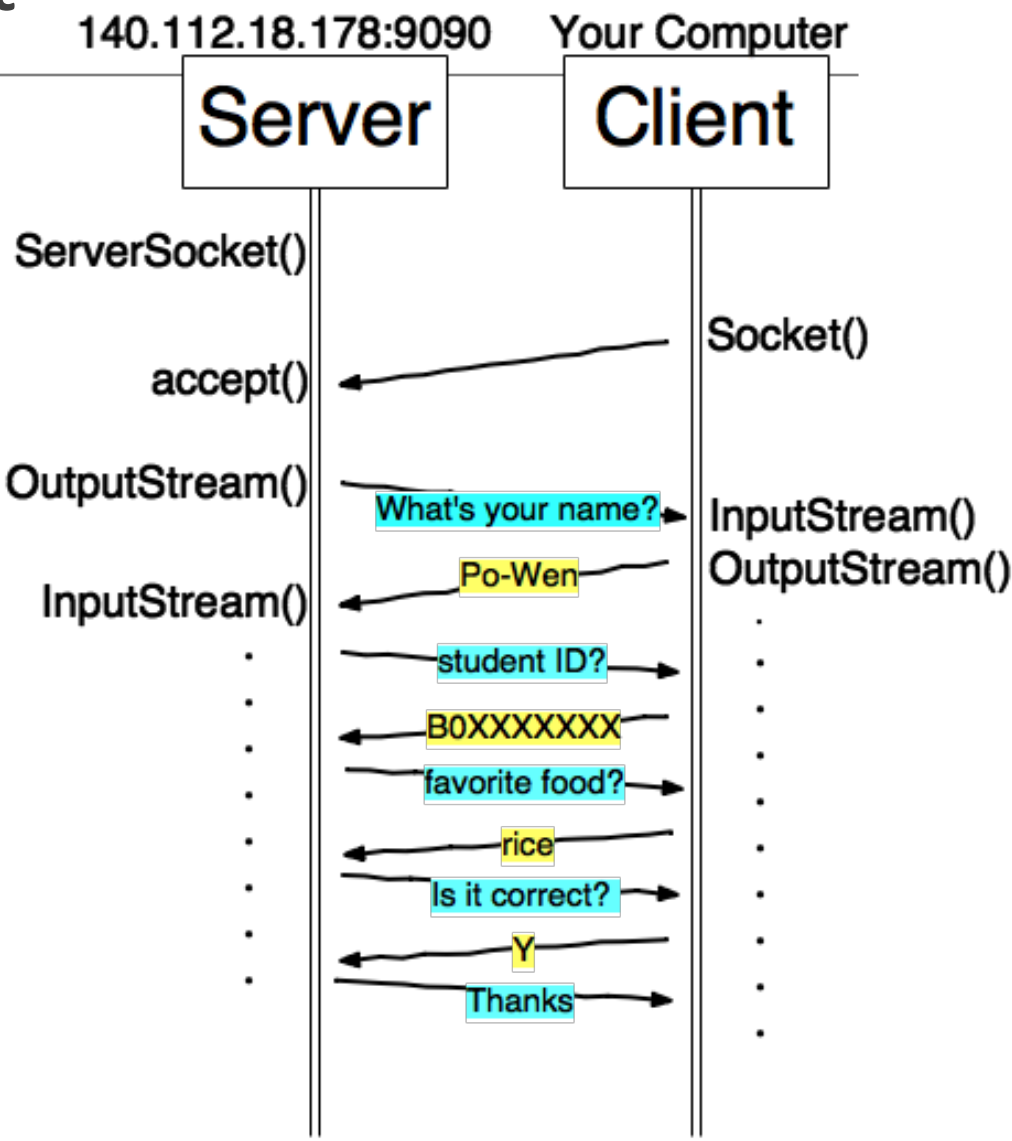


```java
// UDPClient.java
import java.io.*;
import java.nio.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception {
        DatagramSocket clientSocket = new DatagramSocket();

        // Send the sentence to Server 1000 times continously
        InetAddress serverIP = InetAddress.getByName("127.0.0.1");


        for (int i = 1; i <= 10000; ++i) {
            String sentence = "Hello";
            byte[] bytes = sentence.getBytes();

            DatagramPacket sendPkt =
                new DatagramPacket(bytes, bytes.length, serverIP, 9091);
            clientSocket.send(sendPkt);

            /* Suspend for 1us */
            Thread.sleep(1); /* Prevent client side buffer overflow. */
        }

        clientSocket.close();
    }
}
```

```java
// UDPServer.java
import java.io.*;
import java.nio.*;
import java.net.*;

class UDPServer {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(9091);
        System.out.println("Server Ready..");

        DatagramPacket rcvdPkt =
            new DatagramPacket(new byte[128], 128);

        while (true) {
            serverSocket.receive(rcvdPkt);
            String sentence = new String(rcvdPkt.getData());

            System.out.println(sentence);
        }

    }
}
```

HW#2a Assignment

- Practice on TCP/UDP socket programming
- Problem 1: TCP
- Problem2: UDP



For more information

- Visit the Java API, and search “Socket” (or anything else) in the left-bottom panel
  https://docs.oracle.com/javase/8/docs/api/
- The sample codes are from (with a little modification)
  http://www.cse.ust.hk/~muppala/csit5610/labs/Javasock/index.html
- You can refer to the following website for more example
  http://cs.lmu.edu/~ray/notes/javanetexamples/
- GOOD LUCK !