Computer Networks 1
Lab 4

Socket Programming in Java: Chat Application

Names: ....
Student No.: ....

I. Objectives

1. Practice with Socket programming in Java
2. Build a simple chat application using client-server model
3. Multithreaded application in Java

II. Content

1. Socket programming in Java

Consider the following code:

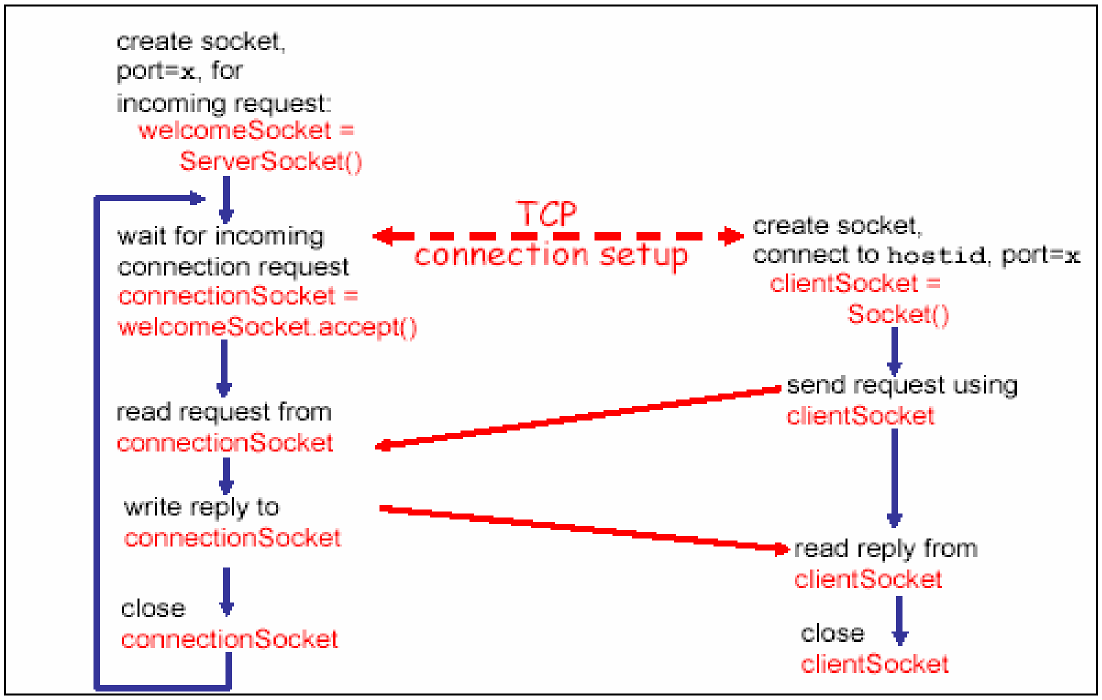
```
try {
      // Get hostname by textual representation of IP address
      InetAddress addr = InetAddress.getByName("127.0.0.1");
      // Get hostname by a byte array containing the IP address
      byte[] ipAddr = new byte[]{127, 0, 0, 1};
      addr = InetAddress.getByAddress(ipAddr);
      // Get the host name
      String hostname = addr.getHostName();
      // Get canonical host name
      String hostnameCanonical = addr.getCanonicalHostName();
}
catch (UnknownHostException e) {
}
```

Exercise 1:

Create a program that connects to a web server and downloads the homepage of this website to local computer.

Ex.: You can test your program with: www.google.com

2. Develop a simple chat application using client-server model



Server application	Client application
Create a server socket	
Listen and wait for incoming connection request	Identify server IP and port
	Create a TCP socket to the server
	Establish a connection to the server
Accept the connection request	
Send/Receive data	Send/Receive data
	Close connection
Close connection	

Exercise 2: Design the user interface for the chat application

3. Multithread in Java

Create an application that has multiple threads running concurrently:

```
class PrimeRun implements Runnable {
   long minPrime;
   PrimeRun ( long minPrime ) {
          this.minPrime = minPrime;
   }
    public void run() {
          // compute primes larger than minPrime
```

```
            . . .
    }
}
PrimeRun p = new PrimeRun(143);
new Thread(p).start();
```

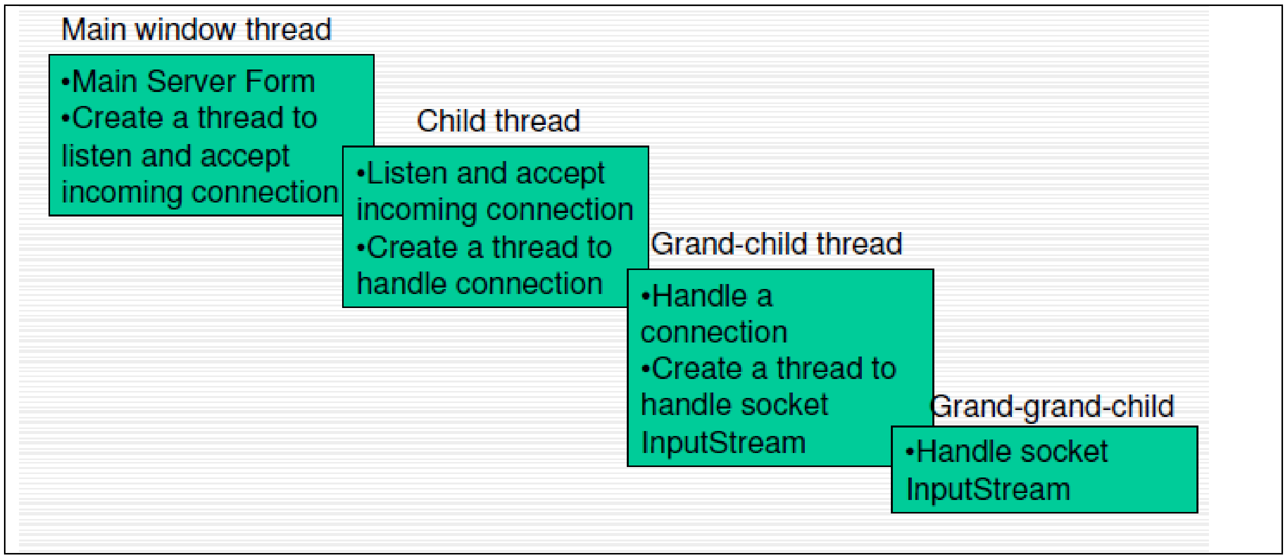
Stop a thread:

Try to use Thread.interrupt() as in the following code and see what happens.

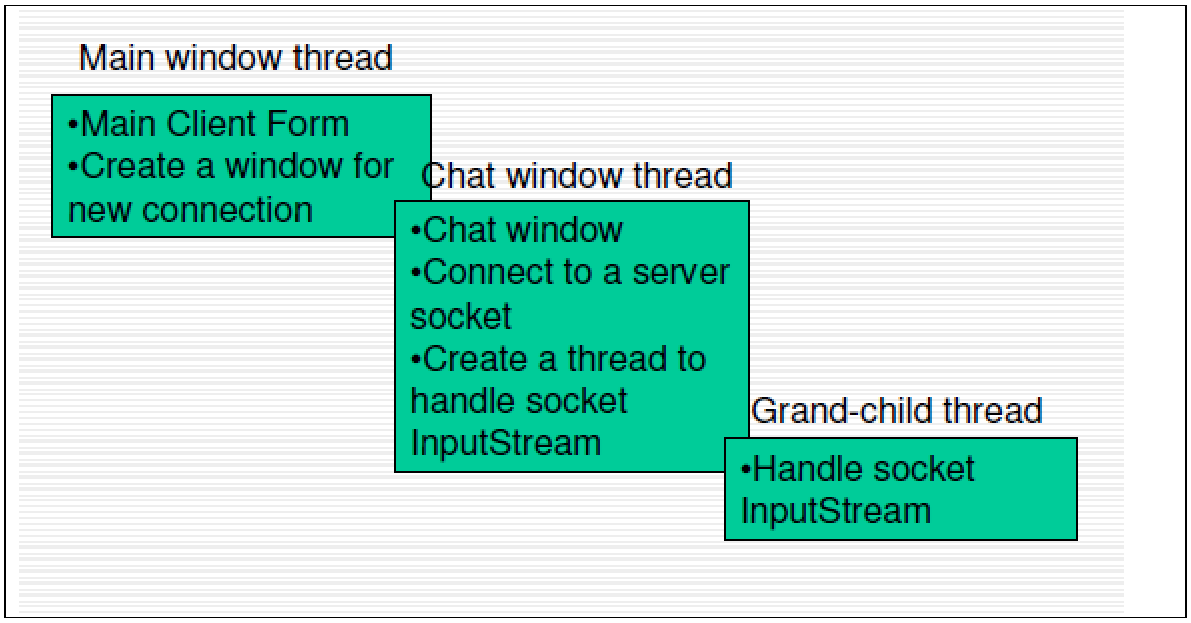
```
public void stop() {
        running = false;
        this.interrupt();
}
public void run() {
        running = true;
        while (running){
                Socket s = socket.accept();
                ...
        }
}
```

```
public void stop() {
        running = false;
        this.interrupt();
}
public void run() {
        running = true;
        while (running){
                Socket s = socket.accept();
                ...
        }
}
```

Multithread in Server Application:


Main window thread
•Main Server Form
•Create a thread to listen and accept incoming connection
Child thread
•Listen and accept incoming connection
•Create a thread to handle connection
Grand-child thread
•Handle a connection
•Create a thread to handle socket InputStream
Grand-grand-child
•Handle socket InputStream

Multithread in Client Application:


Main window thread
•Main Client Form
•Create a window for new connection
Chat window thread
•Chat window
•Connect to a server socket
•Create a thread to handle socket InputStream
Grand-child thread
•Handle socket InputStream

Exercise 3:

Using multithread programming model to make the chat application can talk to many different users concurrently.