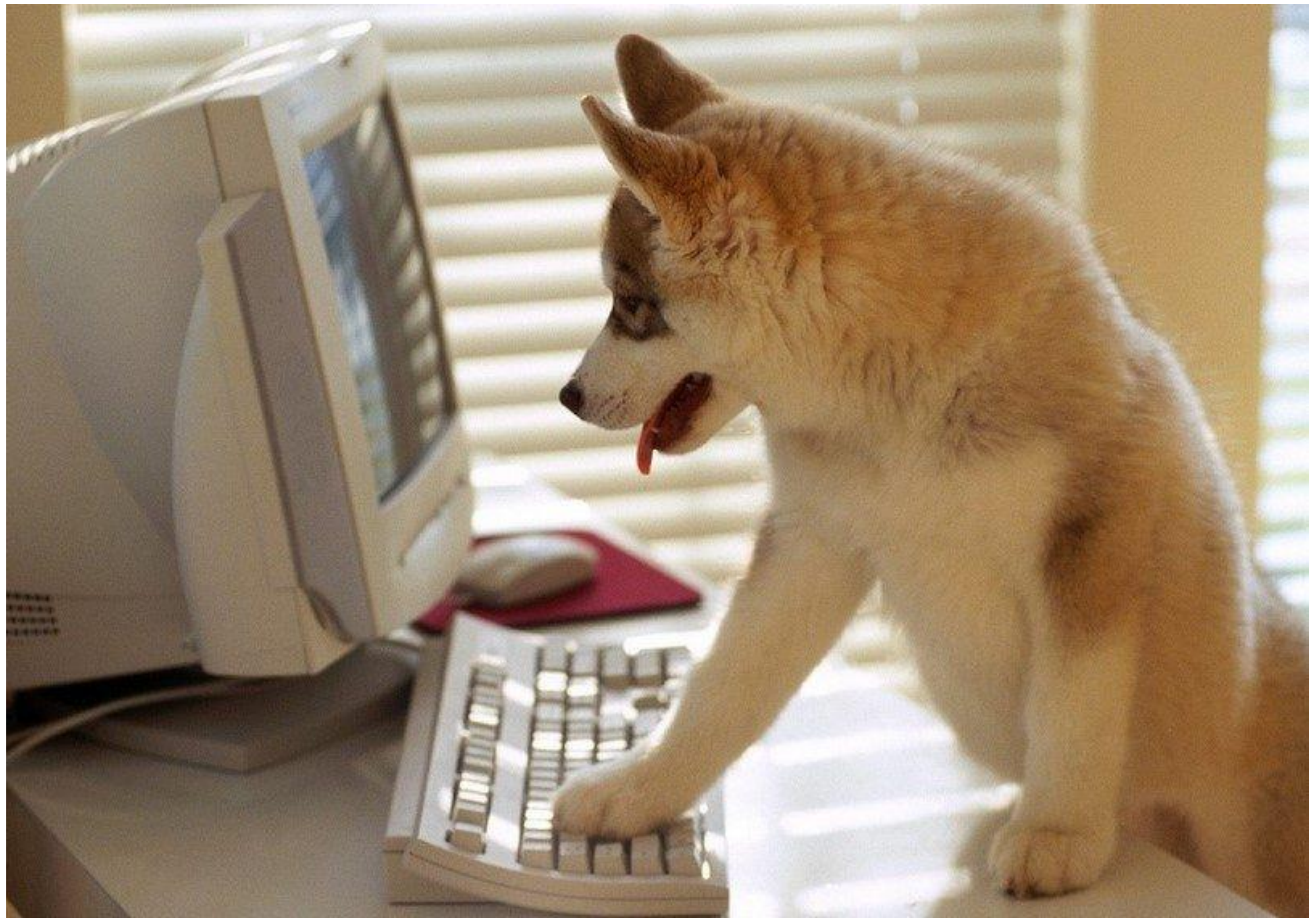
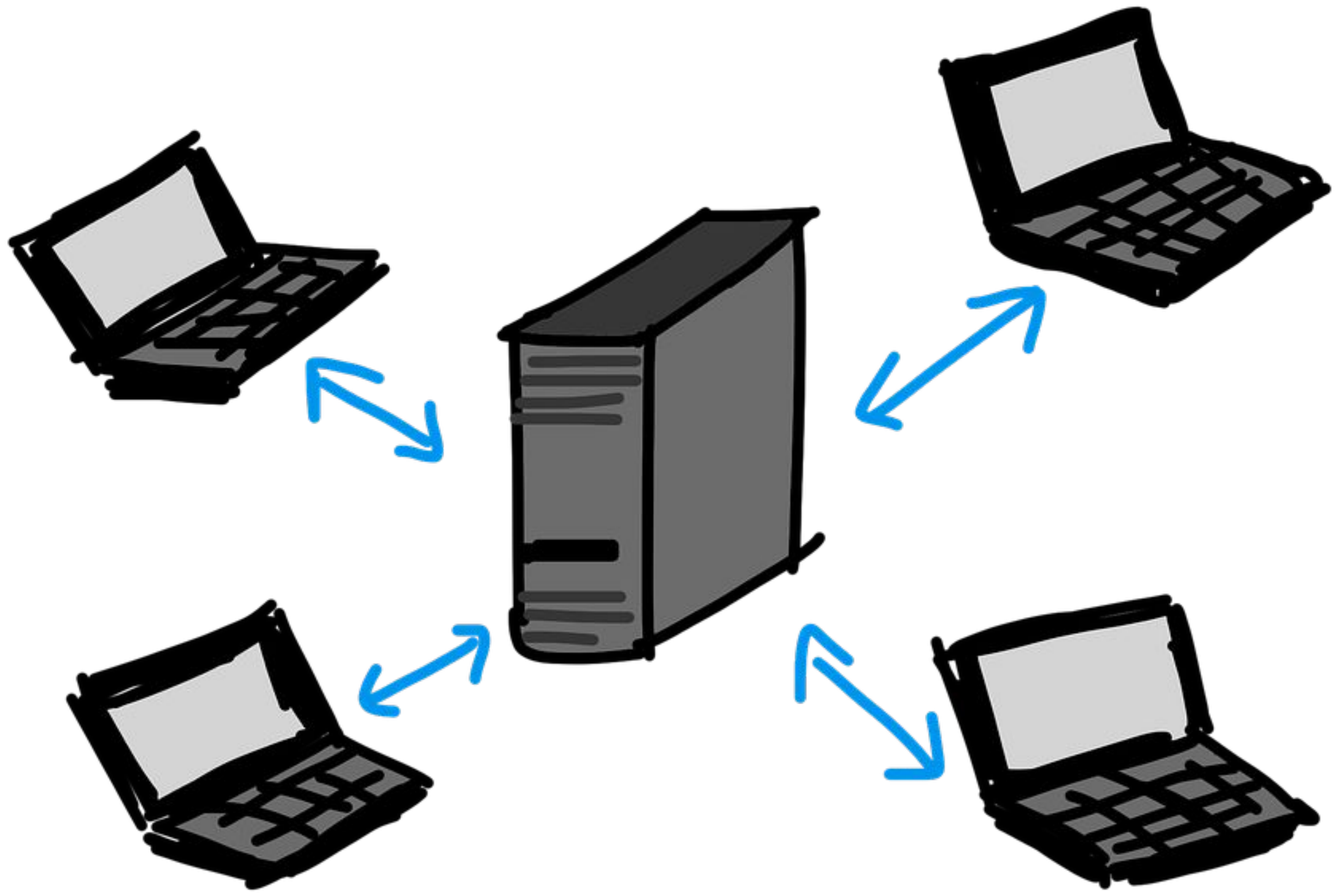
Programming Tutorial [Advanced]



Client Server



Server

NIO:

Channel = transferring data from one channel to another

Server

NIO:

Channel = transferring data from one channel to another

Selector = examines channels and determines which is ready (one Thread for managing multiple channels)

```
Selector selector = Selector.open();
```

Server

NIO:

Channel = transferring data from one channel to another

Selector = examines channels and determines which is ready (one Thread for managing multiple channels)

```
Selector selector = Selector.open();
```

ServerSocketChannel = listens for incoming connections

```
ServerSocketChannel socketChannel = ServerSocketChannel.open();
InetSocketAddress address = new InetSocketAddress(“localhost”, 8080);
socketChannel.bind(address);
socketChannel.configureBlocking(false);
```

Server

NIO:

Channel = transferring data from one channel to another

Selector = examines channels and determines which is ready (one Thread for managing multiple channels)

```
Selector selector = Selector.open();
```

ServerSocketChannel = listens for incoming connections

```
ServerSocketChannel socketChannel = ServerSocketChannel.open();
InetSocketAddress address = new InetSocketAddress(“localhost”, 8080);
socketChannel.bind(address);
socketChannel.configureBlocking(false);
```

SelectionKey = Collection of operations that can be performed

```
SelectionKey selectionKey = socketChannel.register(selector,
                                                   SelectionKey.OP_ACCEPT);
```

Server

```
while (true) {
   selector.select();
   Set<SelectionKey> keys = selector.selectedKeys();
   Iterator<SelectionKey> iterator = keys.iterator();
   while (iterator.hasNext()) {
      SelectionKey myKey = iterator.next();
      if (myKey.isAcceptable()) {
         SocketChannel client = socketChannel.accept();
         client.configureBlocking(false);
         client.register(selector, SelectionKey.OP_READ);
      } else if (myKey.isReadable()) {
         SocketChannel client = (SocketChannel) myKey.channel();
         ByteBuffer buffer = ByteBuffer.allocate(256);
         client.read(buffer);
         try{
            Integer value = Integer.parseInt(new String(buffer.array()).trim());
            result *= value;
            System.out.println("Current result: " + result);
         }catch(NumberFormatException e){
            client.close();
         }
      }
}
```

Server

```
while (true) {
    selector.select();
    Set<SelectionKey> keys = selector.selectedKeys();
    Iterator<SelectionKey> iterator = keys.iterator();
    while (iterator.hasNext()) {
        SelectionKey myKey = iterator.next();
        if (myKey.isAcceptable()) {
            SocketChannel client = socketChannel.accept();
            client.configureBlocking(false);
            client.register(selector, SelectionKey.OP_READ);
        } else if (myKey.isReadable()) {
            SocketChannel client = (SocketChannel) myKey.channel();
            ByteBuffer buffer = ByteBuffer.allocate(256);
            client.read(buffer);
            try{
                Integer value = Integer.parseInt(new String(buffer.array()).trim());
                result *= value;
                System.out.println("Current result: " + result);
            }catch(NumberFormatException e){
                client.close();
            }
        }
}
```

Server

```
while (true) {
    selector.select();
    Set<SelectionKey> keys = selector.selectedKeys();
    Iterator<SelectionKey> iterator = keys.iterator();
    while (iterator.hasNext()) {
        SelectionKey myKey = iterator.next();
        if (myKey.isAcceptable()) {
            SocketChannel client = socketChannel.accept();
            client.configureBlocking(false);
            client.register(selector, SelectionKey.OP_READ);
        } else if (myKey.isReadable()) {
            SocketChannel client = (SocketChannel) myKey.channel();
            ByteBuffer buffer = ByteBuffer.allocate(256);
            client.read(buffer);
            try{
                Integer value = Integer.parseInt(new String(buffer.array()).trim());
                result *= value;
                System.out.println("Current result: " + result);
            }catch(NumberFormatException e){
                client.close();
            }
        }
}
```

Server

```
while (true) {
    selector.select();
    Set<SelectionKey> keys = selector.selectedKeys();
    Iterator<SelectionKey> iterator = keys.iterator();
    while (iterator.hasNext()) {
        SelectionKey myKey = iterator.next();
        if (myKey.isAcceptable()) {
            SocketChannel client = socketChannel.accept();
            client.configureBlocking(false);
            client.register(selector, SelectionKey.OP_READ);
        } else if (myKey.isReadable()) {
            SocketChannel client = (SocketChannel) myKey.channel();
            ByteBuffer buffer = ByteBuffer.allocate(256);
            client.read(buffer);
            try{
                Integer value = Integer.parseInt(new String(buffer.array()).trim());
                result *= value;
                System.out.println("Current result: " + result);
            }catch(NumberFormatException e){
                client.close();
            }
        }
}
```

Client

```
InetSocketAddress address = new InetSocketAddress("localhost", 8080);
SocketChannel client = SocketChannel.open(address);
Scanner scan = new Scanner(System.in);
ArrayList<String> values = new ArrayList<>();
System.out.println("Add your values to multiply, enter 'stop' or 's' when you are done");
String s = scan.nextLine();
while(!s.equals("stop") && !s.equals("s")){
   values.add(s);
   s = scan.nextLine();
}

for (String value : values) {

   byte[] message = new String(value).getBytes();
   ByteBuffer buffer = ByteBuffer.wrap(message);
   client.write(buffer);
   buffer.clear();
   Thread.sleep(2000);
}

client.close();
```

Client

```
InetSocketAddress address = new InetSocketAddress("localhost", 8080);
SocketChannel client = SocketChannel.open(address);
Scanner scan = new Scanner(System.in);
ArrayList<String> values = new ArrayList<>();
System.out.println("Add your values to multiply, enter 'stop' or 's' when you are done");
String s = scan.nextLine();
while(!s.equals("stop") && !s.equals("s")){
   values.add(s);
   s = scan.nextLine();
}

for (String value : values) {

   byte[] message = new String(value).getBytes();
   ByteBuffer buffer = ByteBuffer.wrap(message);
   client.write(buffer);
   buffer.clear();
   Thread.sleep(2000);
}

client.close();
```

Client

```
InetSocketAddress address = new InetSocketAddress("localhost", 8080);
SocketChannel client = SocketChannel.open(address);
Scanner scan = new Scanner(System.in);
ArrayList<String> values = new ArrayList<>();
System.out.println("Add your values to multiply, enter 'stop' or 's' when you are done");
String s = scan.nextLine();
while(!s.equals("stop") && !s.equals("s")){
    values.add(s);
    s = scan.nextLine();
}

for (String value : values) {

    byte[] message = new String(value).getBytes();
    ByteBuffer buffer = ByteBuffer.wrap(message);
    client.write(buffer);
    buffer.clear();
    Thread.sleep(2000);
}

client.close();
```

Client Server

In this course we will use Github Classroom

1. Get a Github Account if you don’t have one
2. Go to: https://classroom.github.com/a/IiXMKrqd (or scan the QR Code with your phone)
3. Authorize Github and accept the assignment
4. Click on the repository