# CS 580 Client-Server Programming
# Fall Semester, 2000
# Doc 5 Java Server Intro
# Contents



## References

Dr. Vinge's CS580 class notes, Spring 2000, http://www-rohan.sdsu.edu/faculty/vinge/courses/spring00/cs580/

Parts of this document are from Dr. Vinge's lecture notes, which in turn based on earlier CS 580 classes taught by Andrew Scherpbier & Roger Whitney

java.net.ServerSocket & Socket. See http://www-rohan.sdsu.edu/doc/java/jdk1.2/docs/api/java/net/package-summary.html or http://java.sun.com/products/jdk/1.2/docs/api/java/net/package-summary.html

## Reading

java.net.ServerSocket.
java.net.Socket

# What is a Server?

## Server

Any program that waits for incoming communication requests from a client

Extracts requested information from data and return to client

Basic algorithm:

```
while (true) {
  Wait for an incoming request;
  Perform whatever actions are requested;
}
```

# Some Basic Server Issues

- How to wait for an incoming request?
- How to know when there is a request?
- What happens when there are multiple requests?
- How do clients know how to contact server?
- How to parse client request?
- How do we know when the server has the entire request?

# Java TCP Sockets

## ServerSocket basic methods

public ServerSocket(int port) //port = 0 gives random port

public Socket accept() throws IOException
public void close() throws IOException
public int getLocalPort()

## Socket basic methods

public InputStream getInputStream() throws IOException
public OutputStream getOutputStream() throws IOException

# A Simple Server

```
import java.net.Socket;
import java.net.ServerSocket;
import java.io.*;
import java.util.Date;

class SimpleDateServer {

   public static void main(String[] args) throws IOException {
      ServerSocket acceptor = new ServerSocket(0);
      System.out.println("On port " + acceptor.getLocalPort());

      while (true) {
         Socket client = acceptor.accept();
         processRequest(
            client.getInputStream(),
            client.getOutputStream());
         client.close();
      }
   }
```

(Why "new ServerSocket(0)"?)

# Processing Client Request

```
  static void processRequest(InputStream in,OutputStream out)
    throws IOException {

    BufferedReader parsedInput =
          new BufferedReader(new InputStreamReader(in));

    // the "true" is to get autoflushing:
    PrintWriter parsedOutput = new PrintWriter(out,true);

    String inputLine = parsedInput.readLine();

    if (inputLine.startsWith("date")) {
      Date now = new Date();
      parsedOutput.println(now.toString());
    }
  }
}
```

Note: This server is just a first example. It needs a lot of work. We will be working on improving it in later lectures.

# Running the Server

Sample run of SimpleDateServer.
(I typed everything appearing in bold font here.)

rohan 16-> **java SimpleDateServer &**
[1] 16269
On port 62047

rohan 17-> **telnet rohan 62047**
Trying 130.191.3.100...
Connected to rohan.sdsu.edu.
Escape character is '^]'.
**date today**
Mon Sep 04 13:37:30 PDT 2000
Connection closed by foreign host.

rohan 18-> **telnet rohan 62047**
Trying 130.191.3.100...
Connected to rohan.sdsu.edu.
Escape character is '^]'.
**time**
Connection closed by foreign host.

In this class, shut things down:

rohan 19-> fg
java SimpleDateServer
^C

## Simple Server Issues

- How do we write tests for our server?

  Automate testing of method processRequest()
  Test main by hand (for now)

- Request processing blocks any other connections.

Using our SimpleDateServer

> Client A builds connection to server,
> Client A goes to lunch
> Client B builds connection to server and ... :-(

Solution:
Multiple connections need to be accepted concurrently.

More on this in later lectures.