



# The Internet

Chris Piech  
CS106A, Stanford University

I came here to learn to  
program the internet...



For the third time ever in  
CS106A:

# Learning Goals

1. Write a program that can make internet requests
2. Write a program that can respond to internet requests



How does your phone  
communicate with facebook?

The Java program on your  
**phone** talks to the Java  
program at **Facebook**

Face Book Server



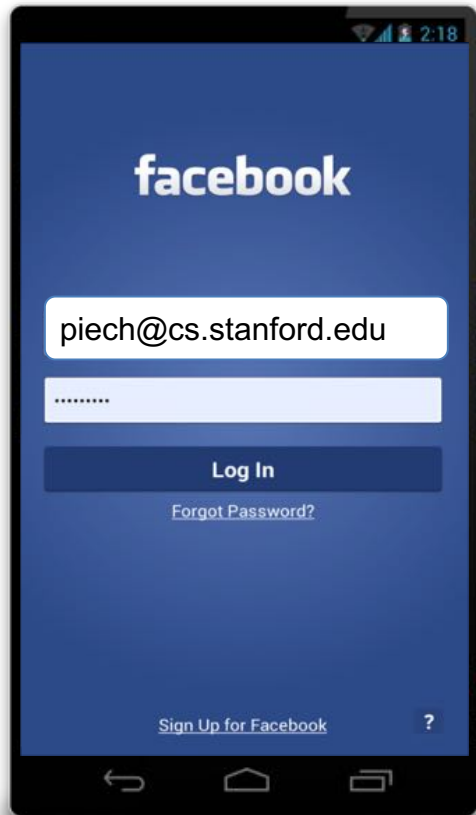
\* Android phones run Java. So do facebook servers





# Face Book Server

Is this legit?

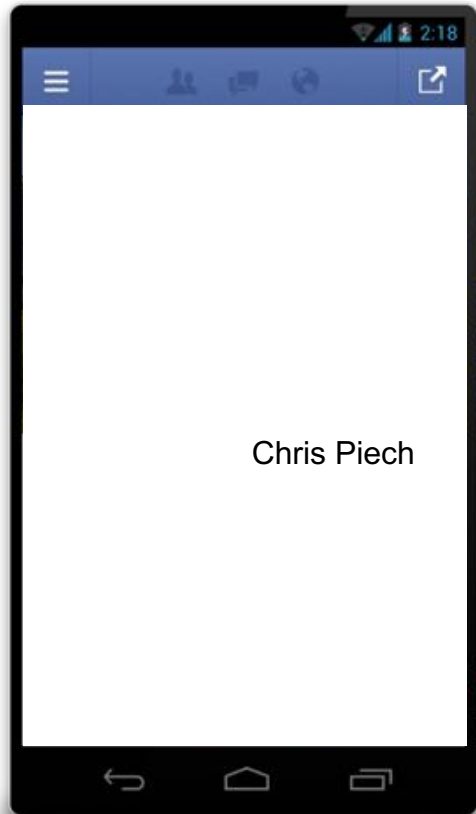


piech@cs.stanford.edu  
is now logged in



# Face Book Server

Send me the **full name** for  
piech@cs.stanford.edu



"Chris Piech"



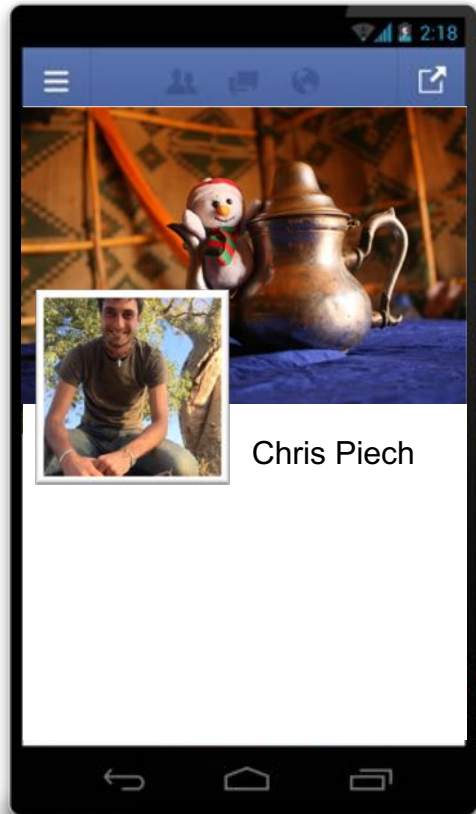
# Face Book Server

Send me the **cover photo** for  
piech@cs.stanford.edu



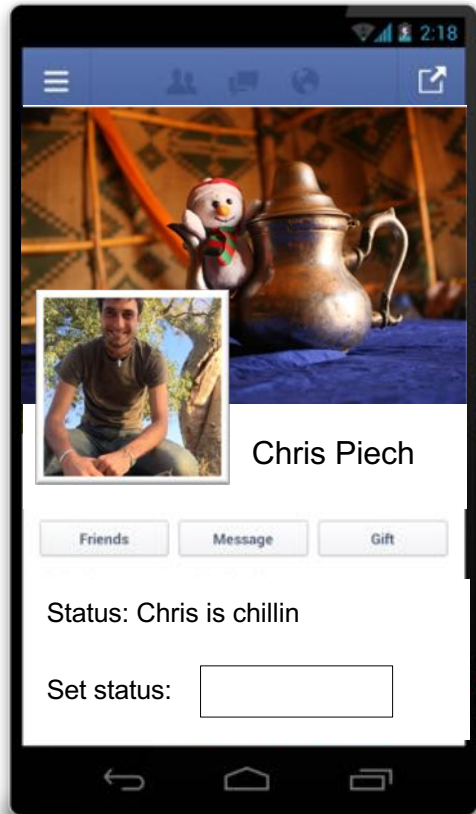
# Face Book Server

Send the **profile photo** for  
piech@cs.stanford.edu



# Face Book Server

Send the **status** for  
piech@cs.stanford.edu

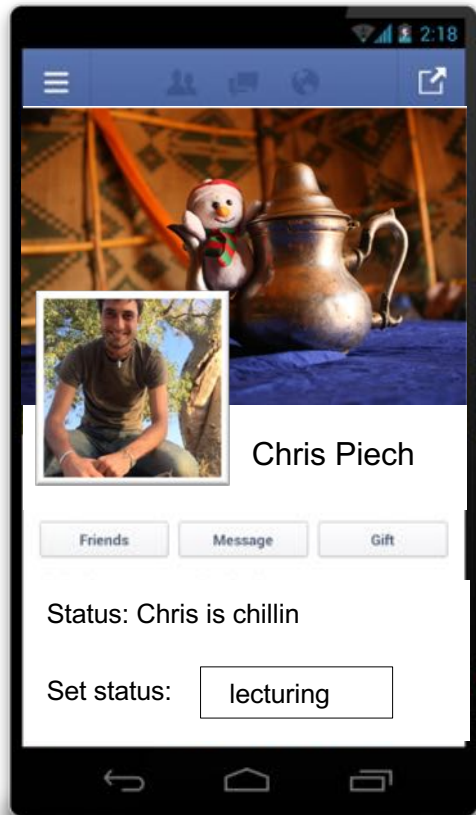


"chillin"



Set the **status** for  
piech@cs.stanford.edu  
to be **"lecturing"**

Face Book Server

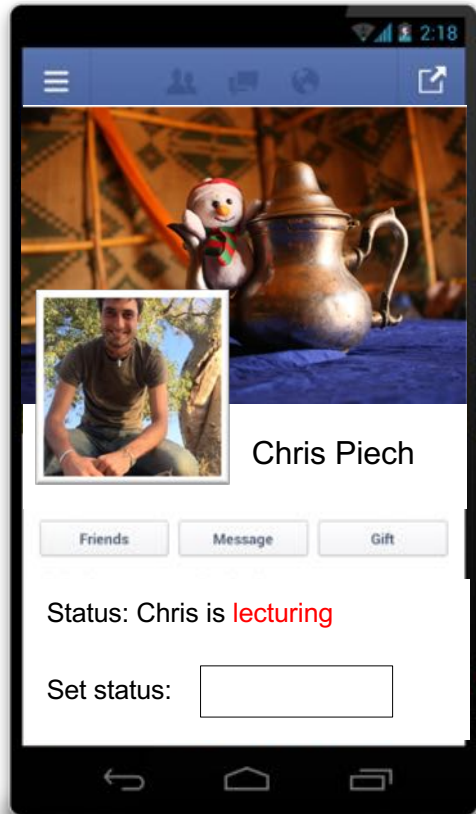


"success"



Send me the **status** for  
piech@cs.stanford.edu

Face Book Server



"lecturing"



# Background: The Internet



The internet is just many programs sending messages (as *Strings*)

Thanks Nick for the teaching YEAH





# Background: The Internet



The internet is just many programs sending messages (as *Strings*)



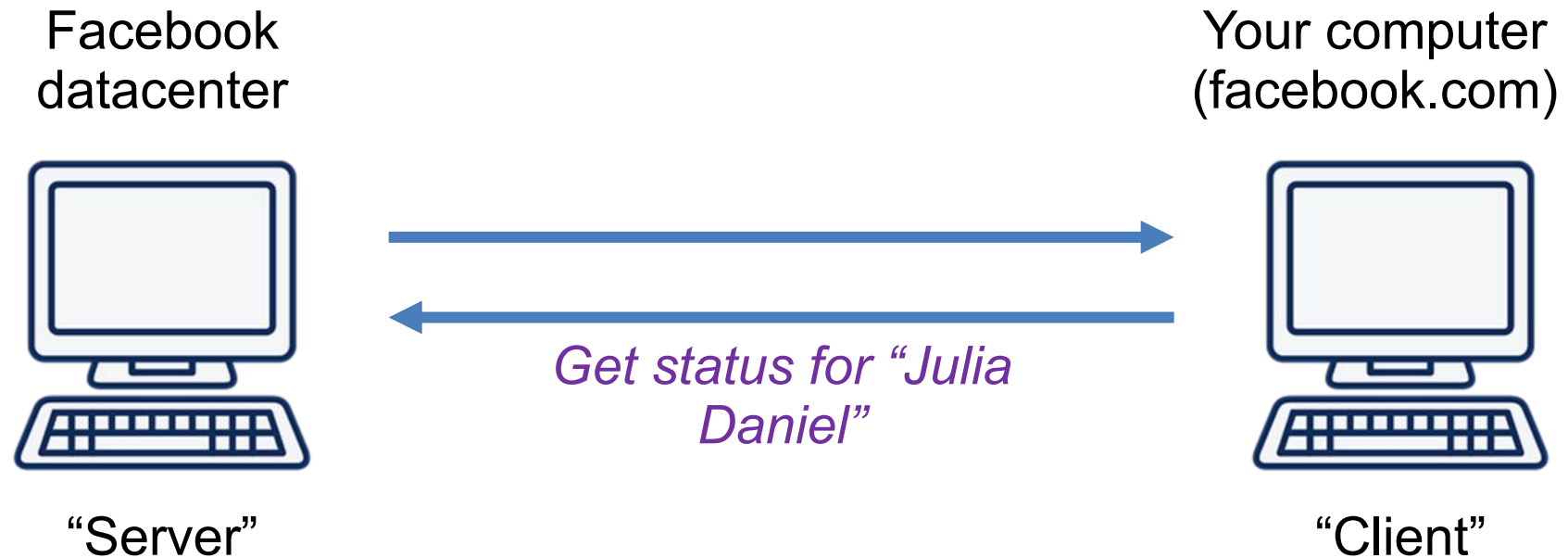
# Background: The Internet



The internet is just many programs sending messages (as *Strings*)



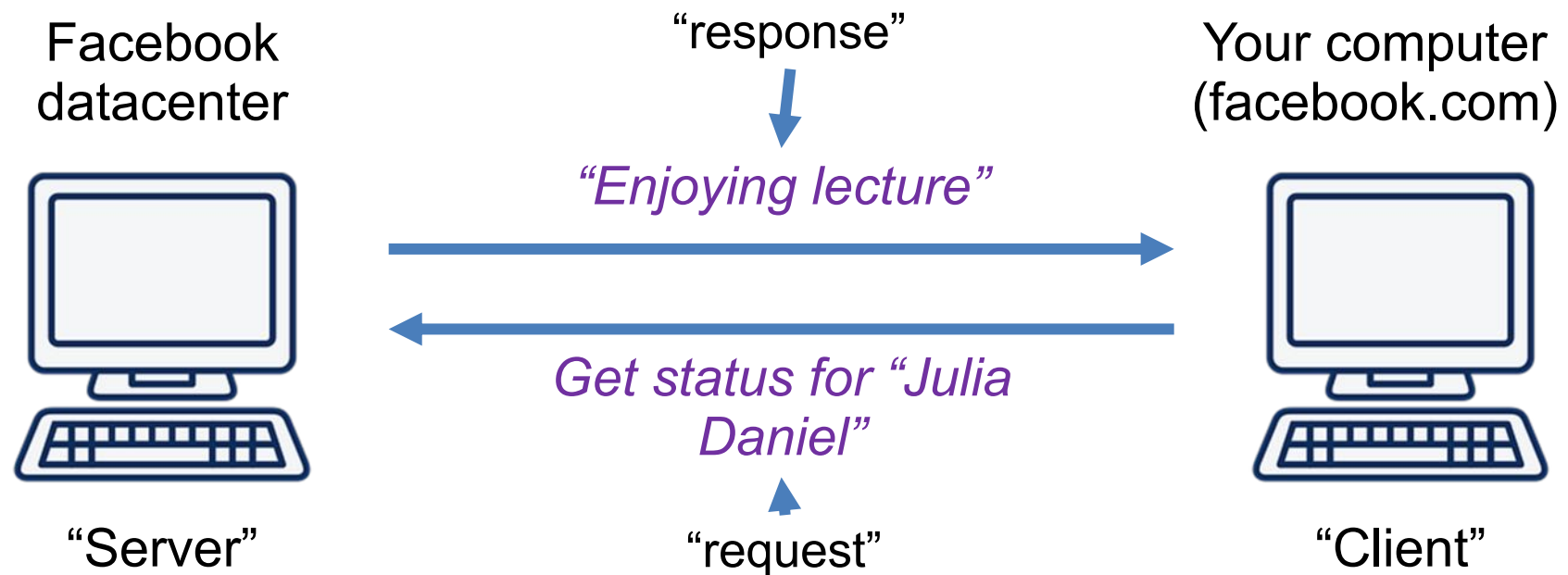
# Background: The Internet



The internet is just many programs sending messages (as *Strings*)



# Background: The Internet



The internet is just many programs sending messages (as *Strings*)





There are two types of  
internet programs. Servers  
and Clients



# Internet 101

# Servers are computers (running code)

Face Book Server



=

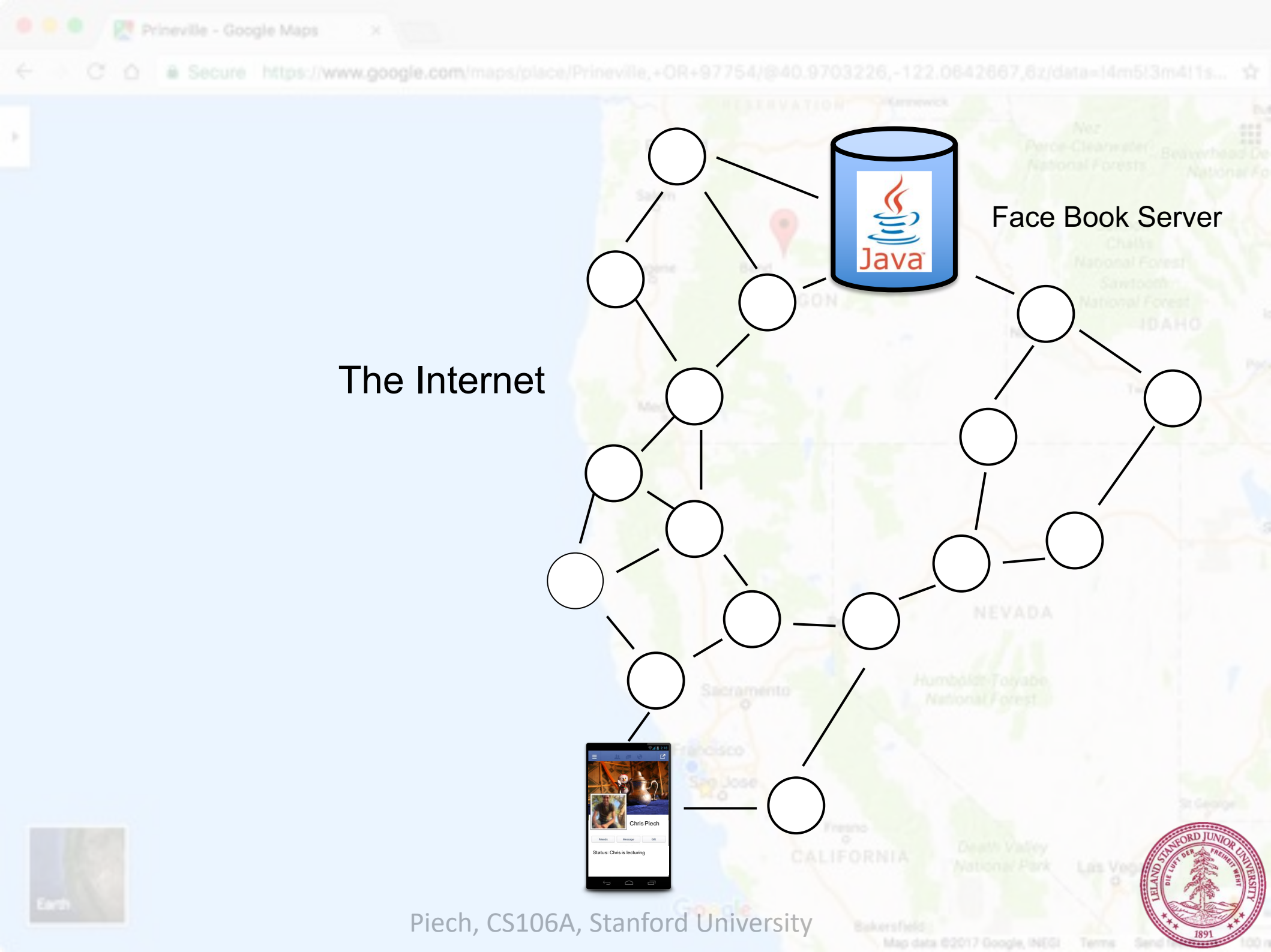


Facebook's closest  
datacenter is here

I am here



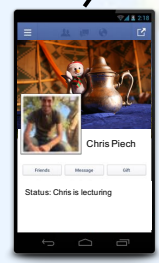


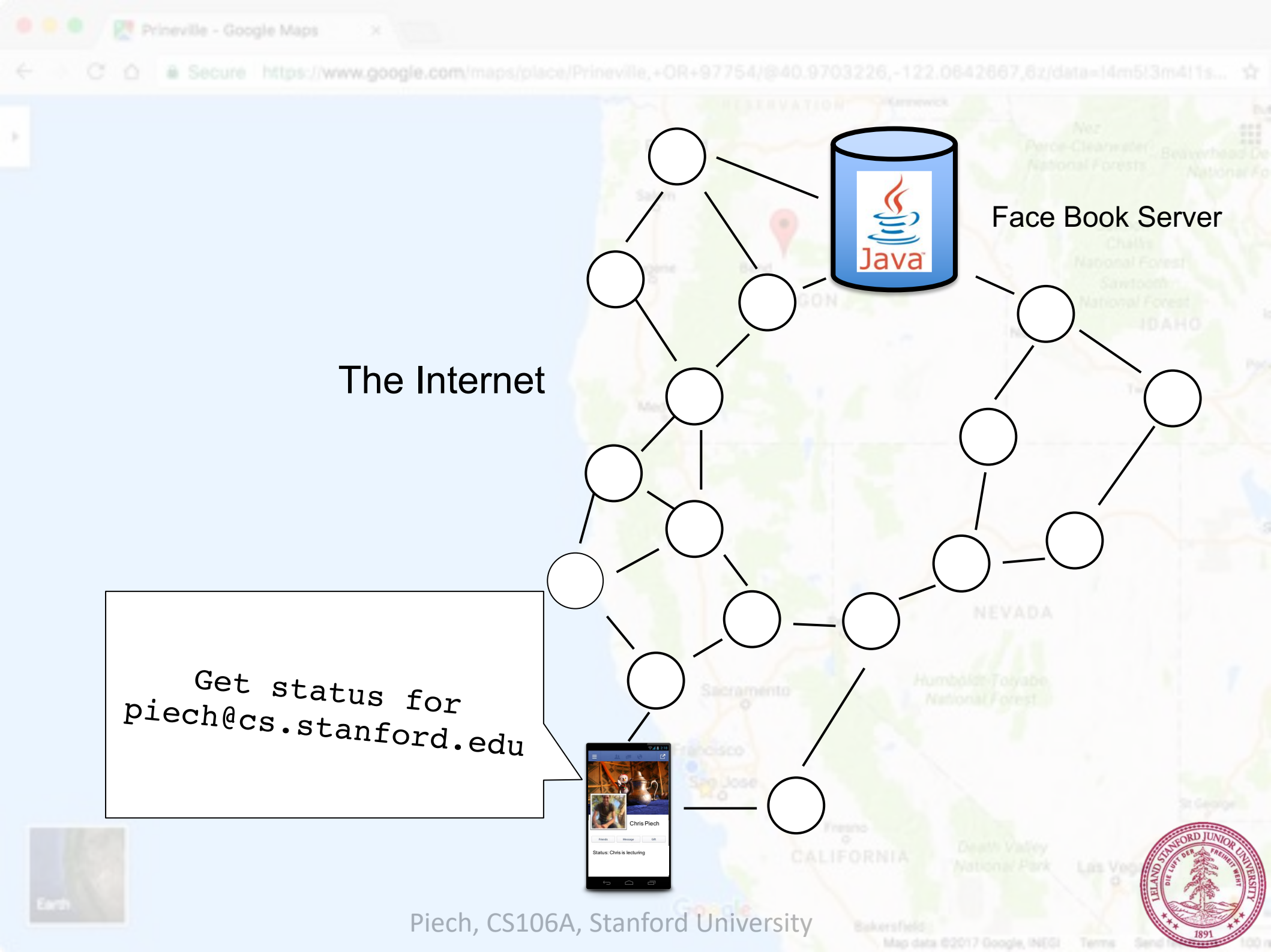


The Internet



Face Book Server



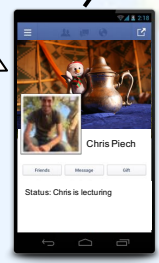


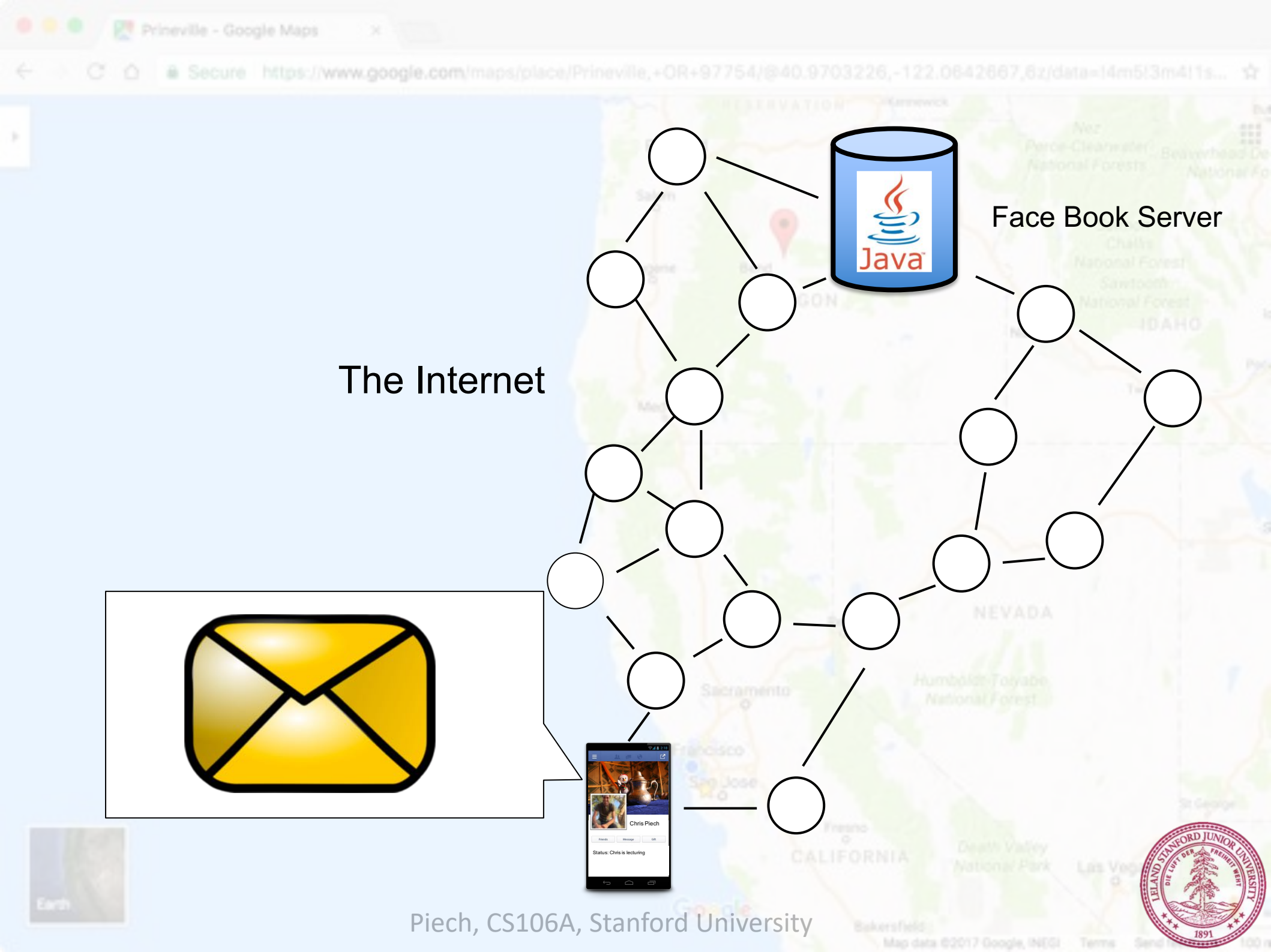
The Internet



Face Book Server

Get status for  
piech@cs.stanford.edu

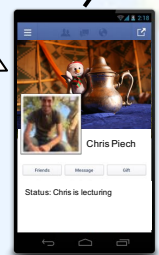
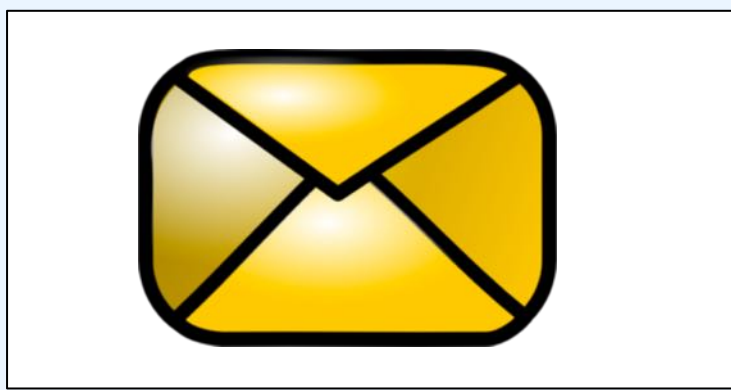


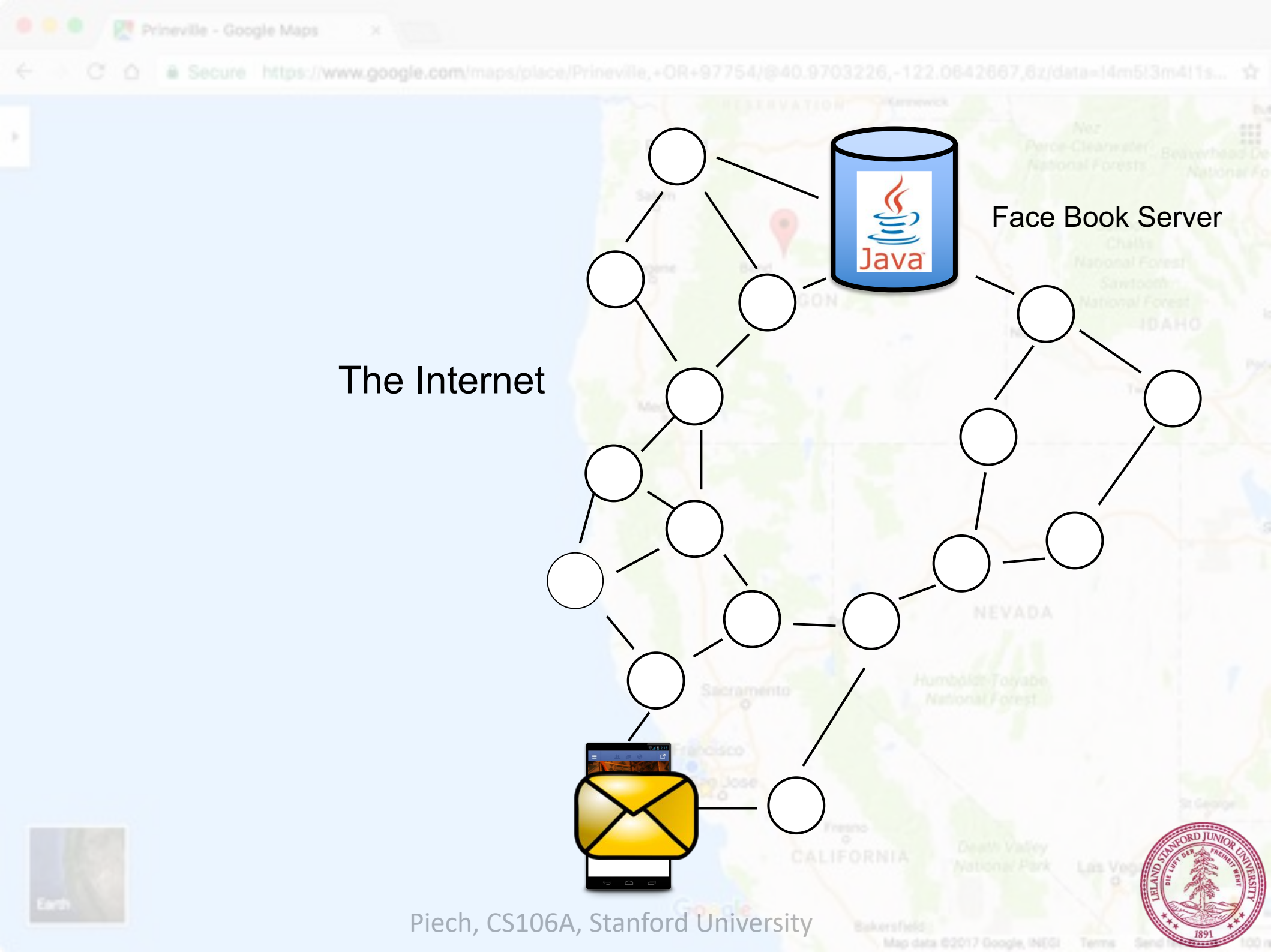


The Internet



Face Book Server

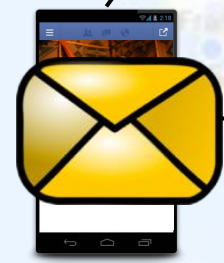




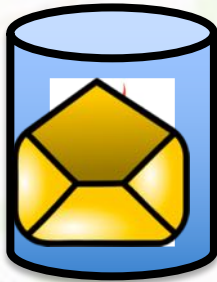
The Internet



Face Book Server

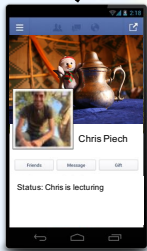
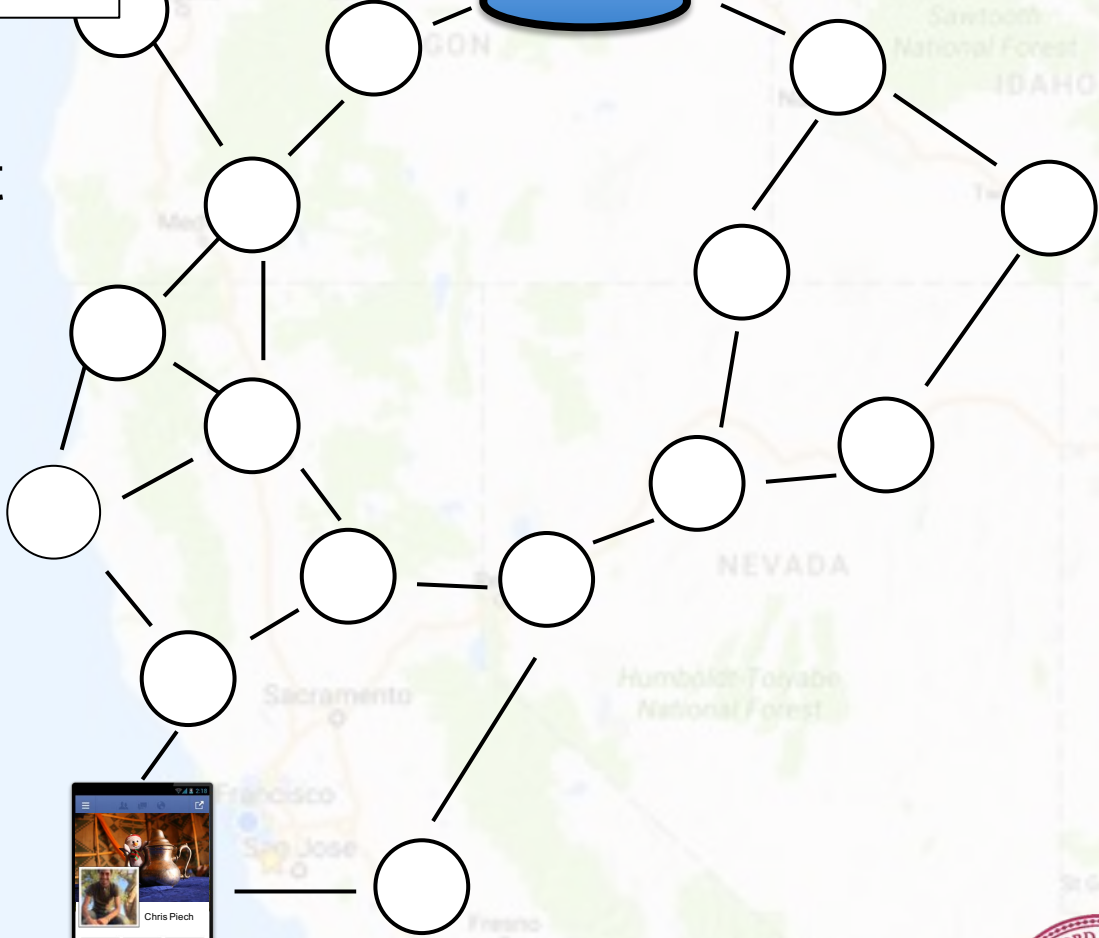


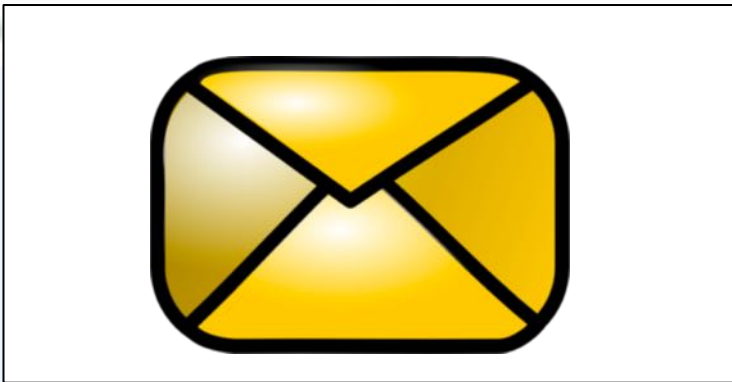
teaching



Face Book Server

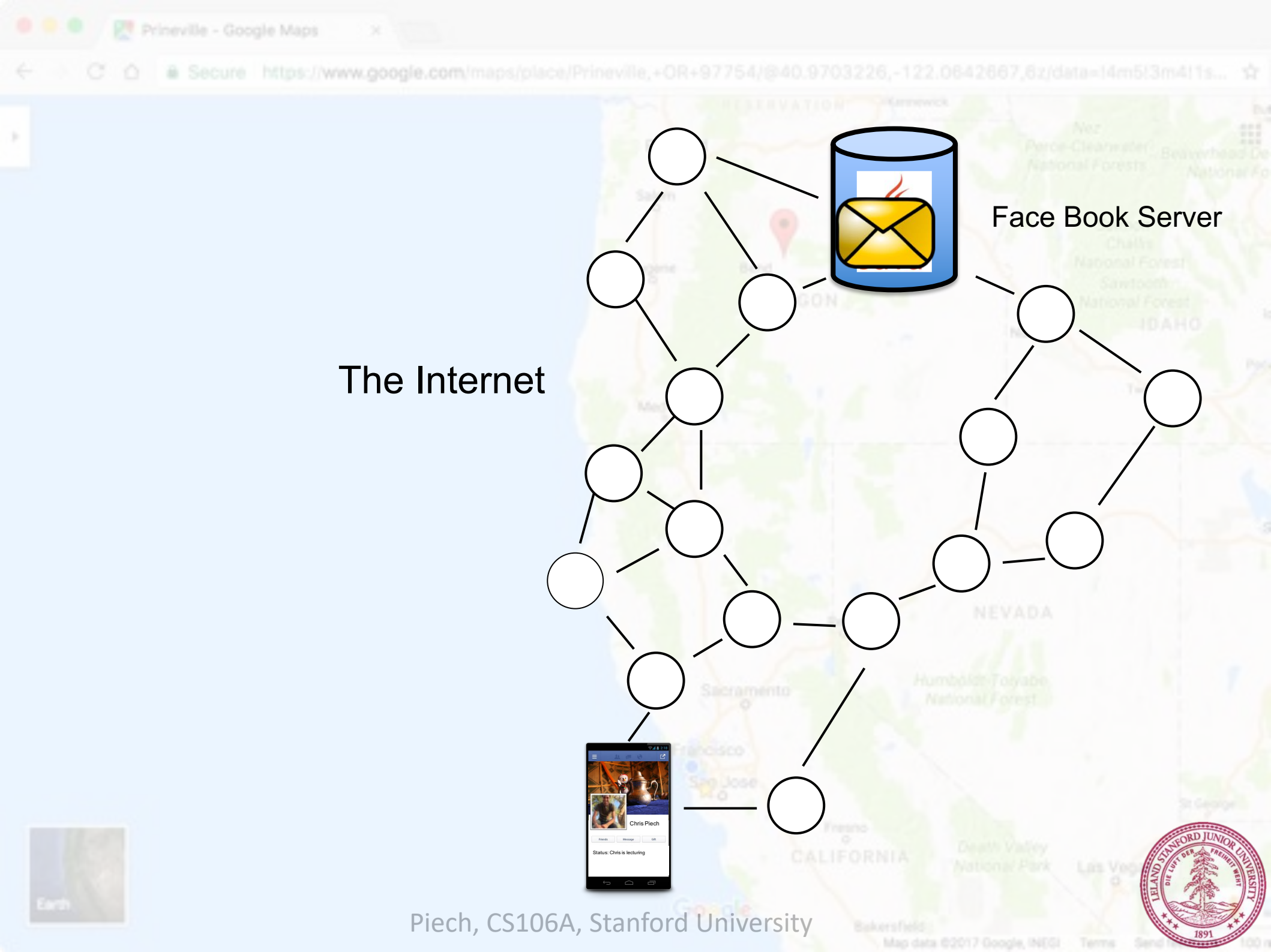
The Internet





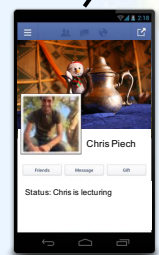
The Internet





The Internet

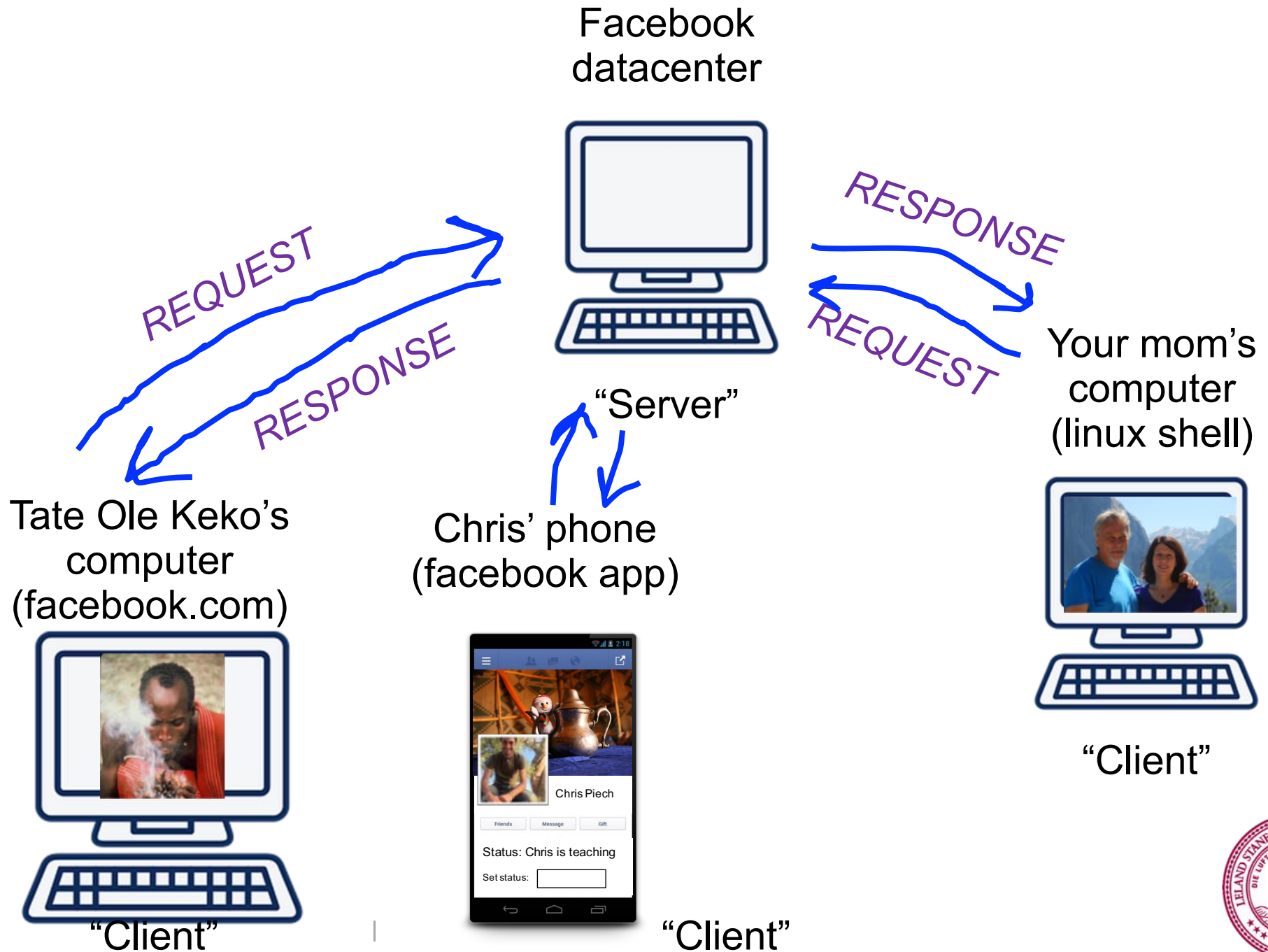
Face Book Server



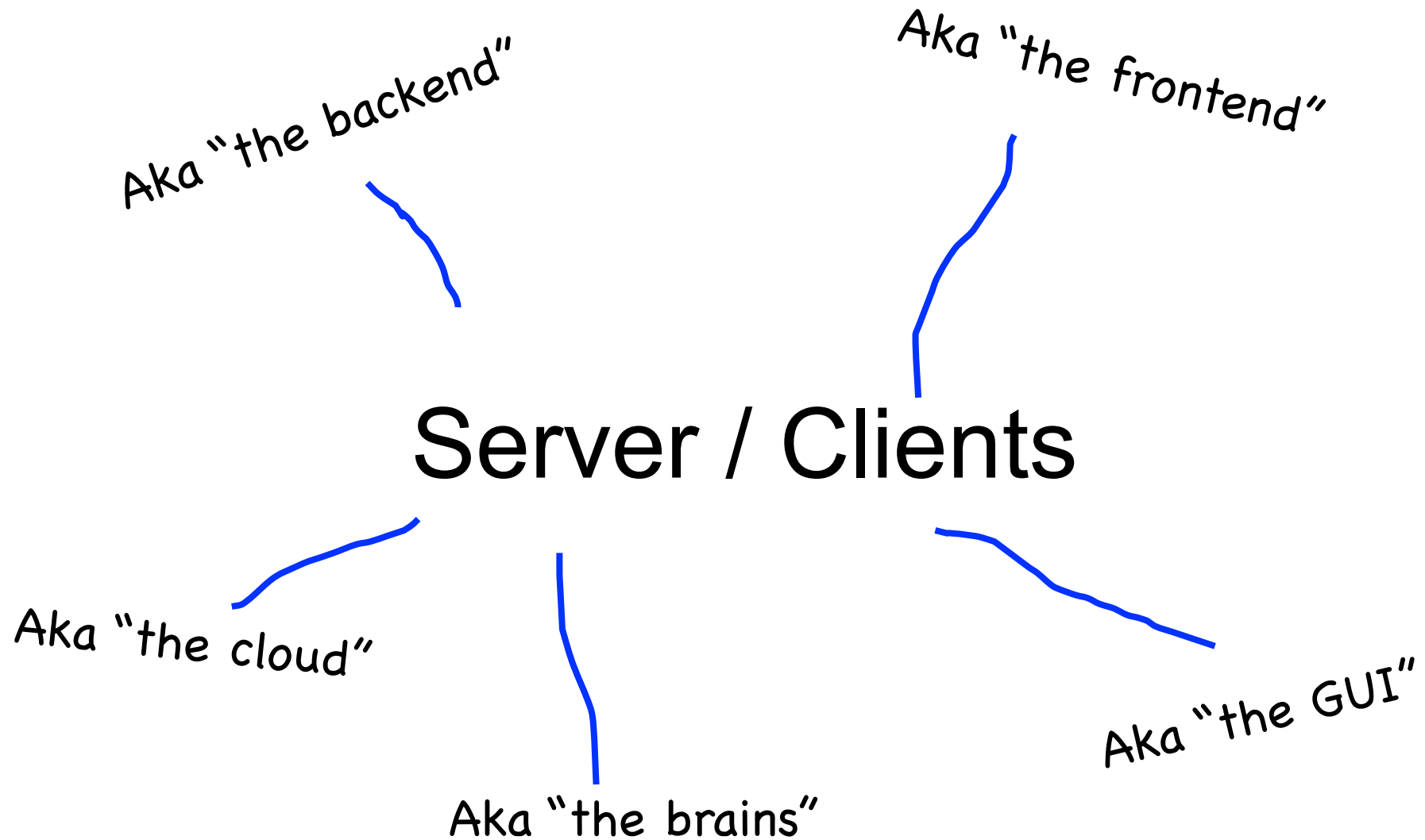
Many computers can connect  
to the same server



# The Internet



# Most of the Internet





There are two types of  
internet programs. Servers  
and Clients

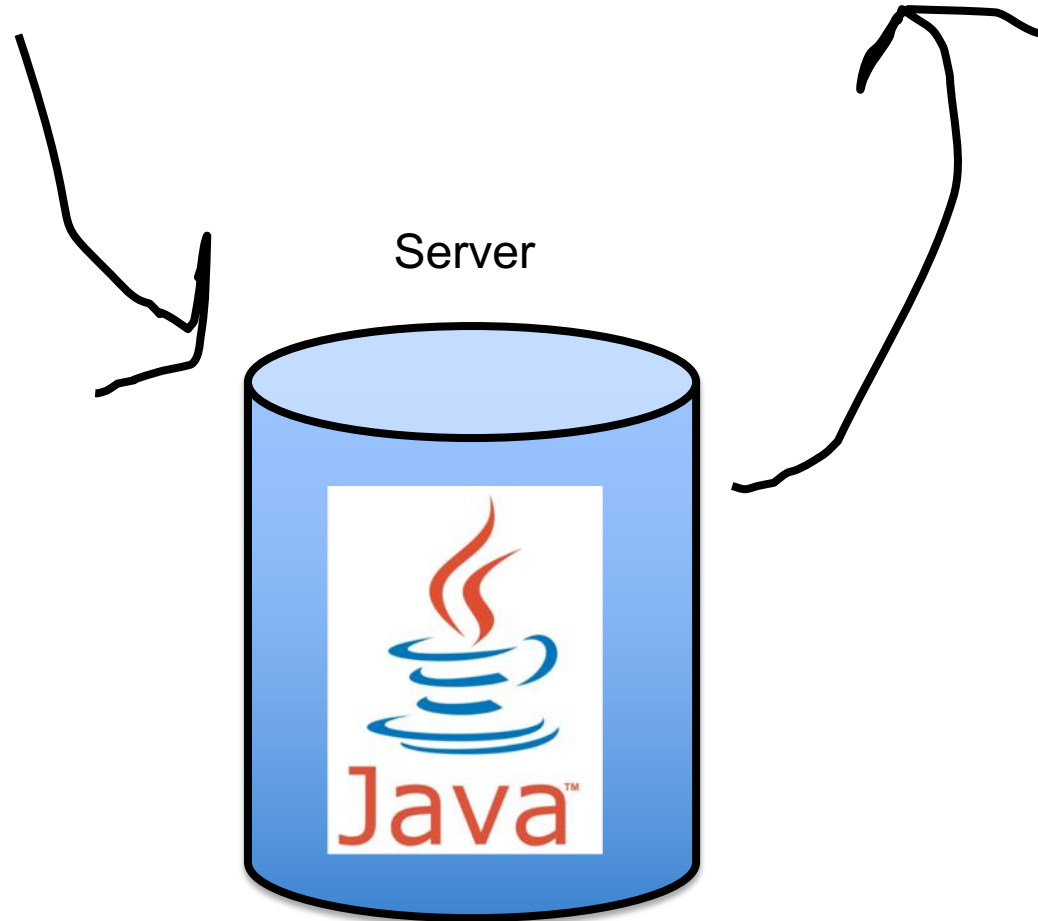


First, the server

# A Server's Simple Purpose

**Request**  
`someRequest`

**String**  
`serverResponse`



# A Server's Simple Purpose

**Request**  
someRequest

**String**  
serverResponse

```
ChatServer
Starting server on port 8080...
getMsgs
newMsg
Added new message
getMsgs
Returned 1 messages
getMsgs
Returned 1 messages
newMsg
Added new message
getMsgs
Returned 1 messages
getMsgs
```



# Servers on one slide

1

```
public String requestMade(Request request) {  
    // server code goes here  
}
```

2

```
// make a Server object  
private SimpleServer server  
    = new SimpleServer(this, 8000);
```

3

```
public void run(){  
    // start the server  
    server.start();  
}
```



# A Server's Simple Purpose

1

```
public String requestMade(Request request) {  
    // server code goes here  
}
```

2

```
// make a Server object  
private SimpleServer server  
    = new SimpleServer(this, 8000);
```

3

```
public void run(){  
    // start the server  
    server.start();  
}
```





# What is a Request?



```
/* Request has a command */  
String command;
```

```
/* Request has parameters */  
HashMap<String,String> params;
```

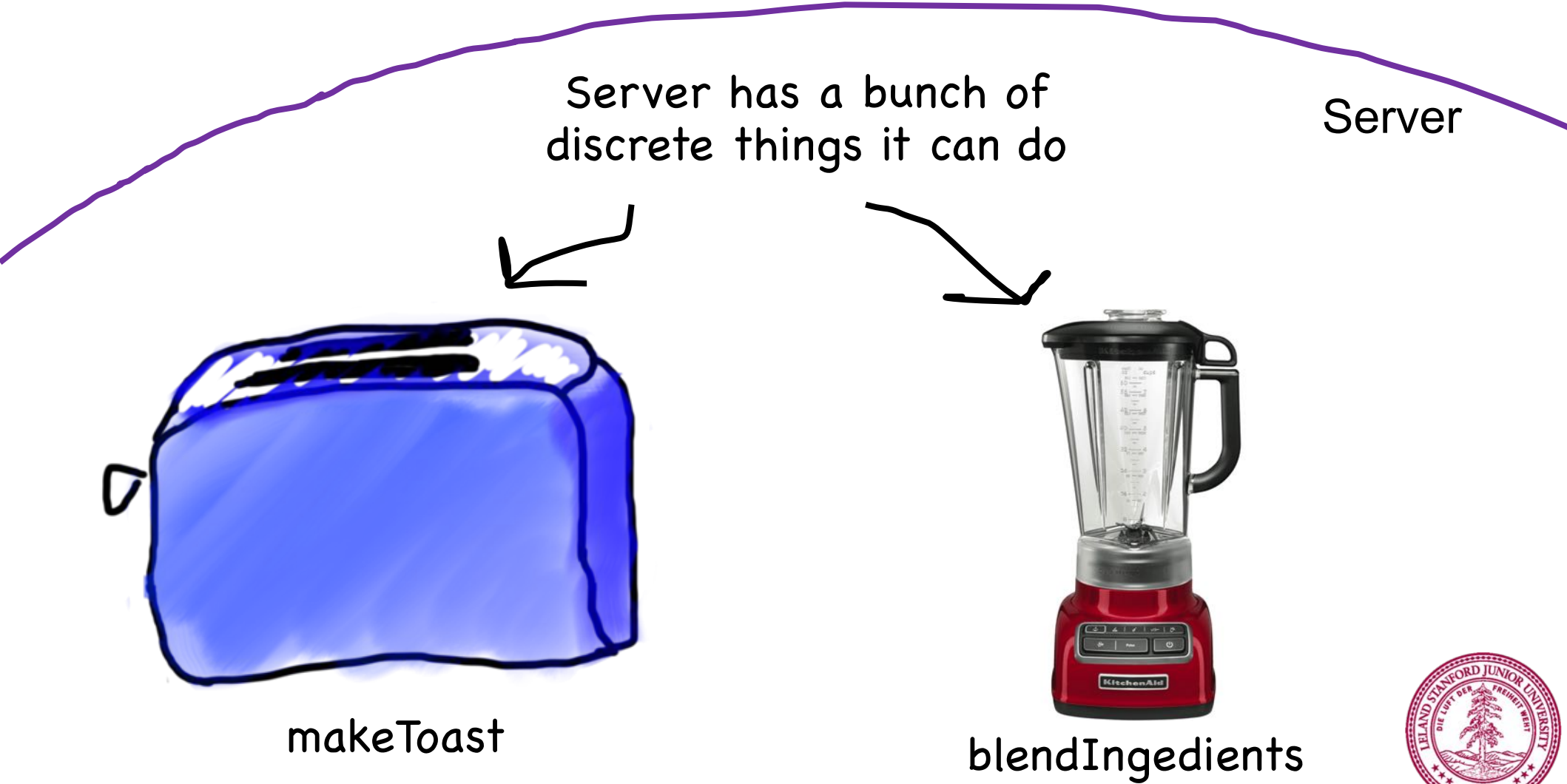
Request request

---

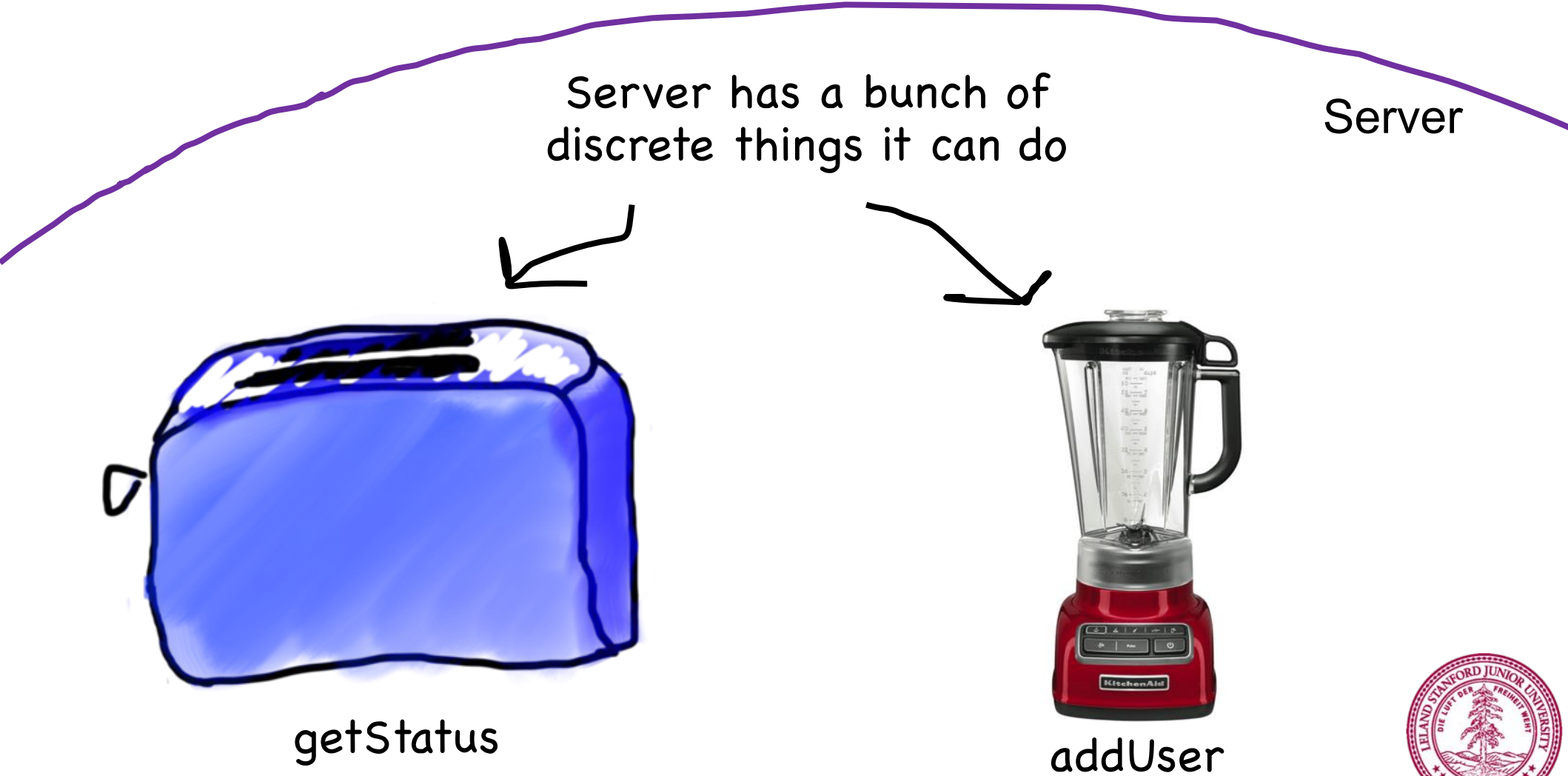
```
// methods that the server calls on requests  
request.getCommand();  
request.getParam(key); //returns associated value
```



# Requests are like Remote Method Calls

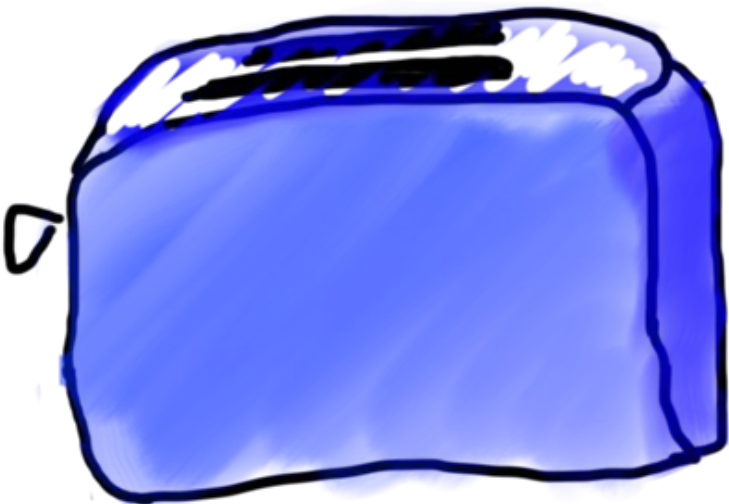


# Requests are like Remote Method Calls



# Requests are like Remote Method Calls

Server



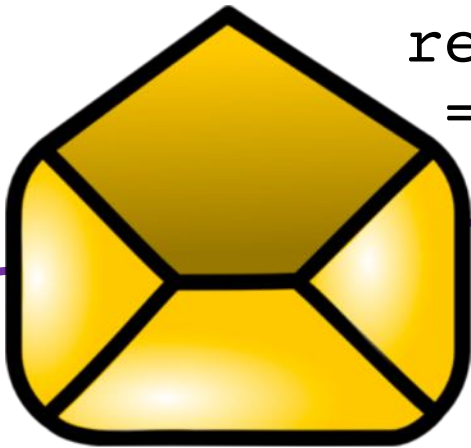
getStatus



addUser



# Requests are like Remote Method Calls



```
request.getCommand();  
=> "getStatus"
```

Server



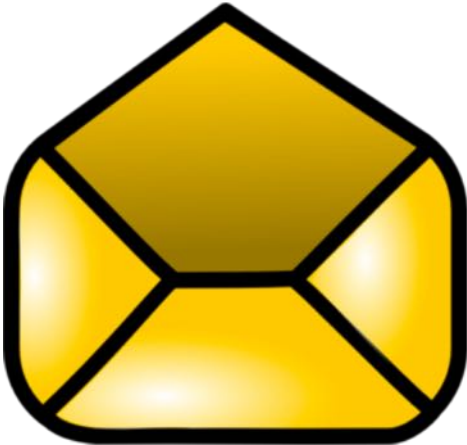
getStatus



addUser



# Requests are like Remote Method Calls



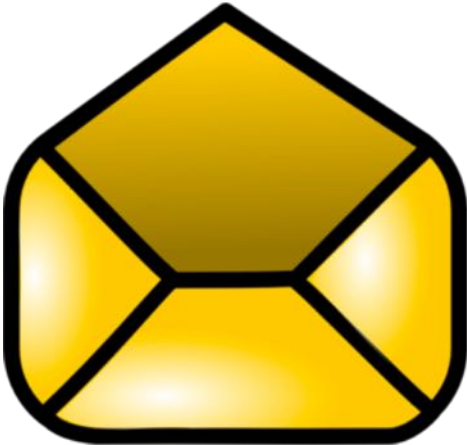
To make toast, I need a parameter which is the kind of bread



getStatus



# Requests are like Remote Method Calls



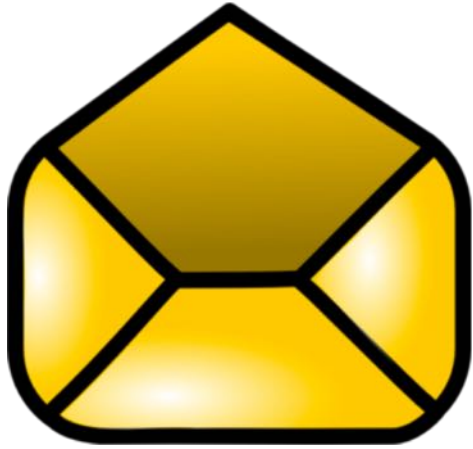
I was given a parameter!



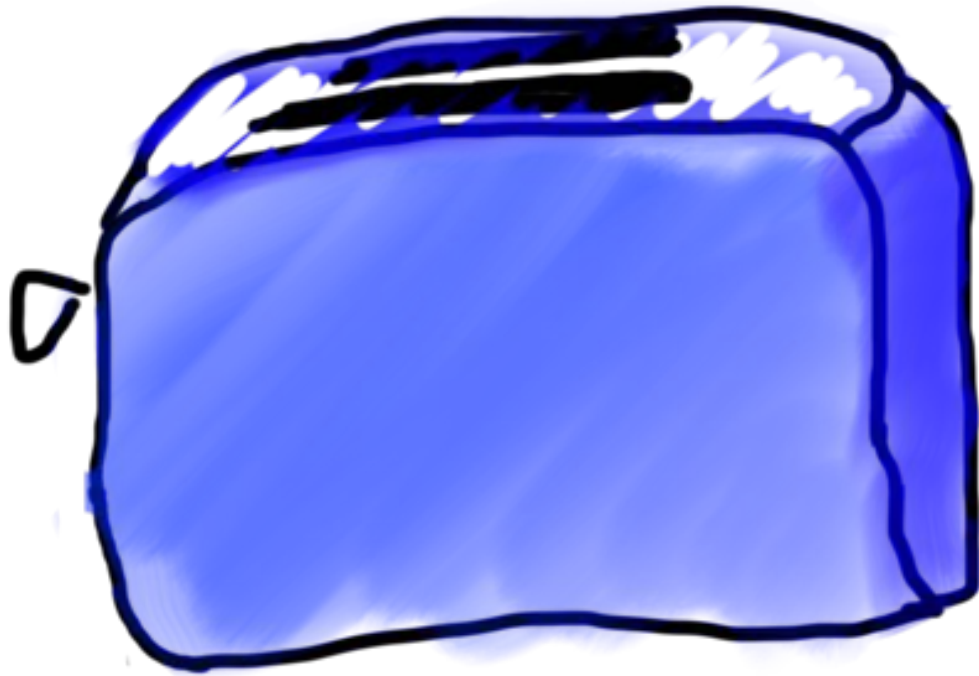
getStatus



# Requests are like Remote Method Calls



```
request.getParam("userName")
```

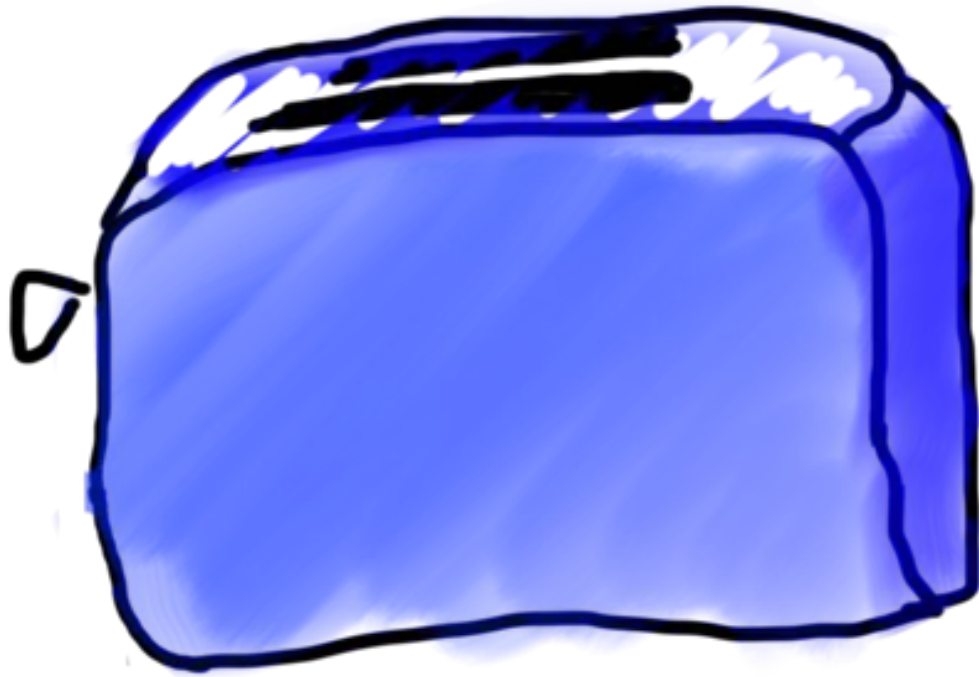
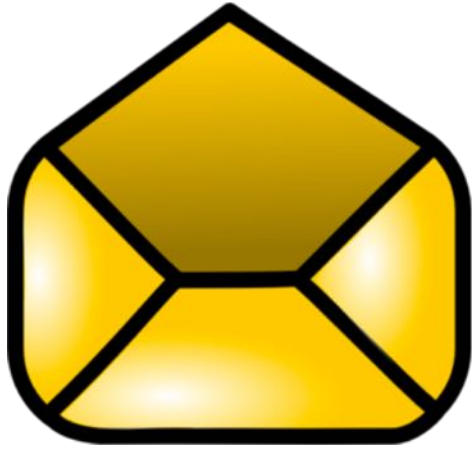


```
getStatus
```





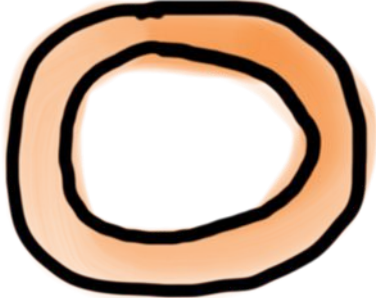
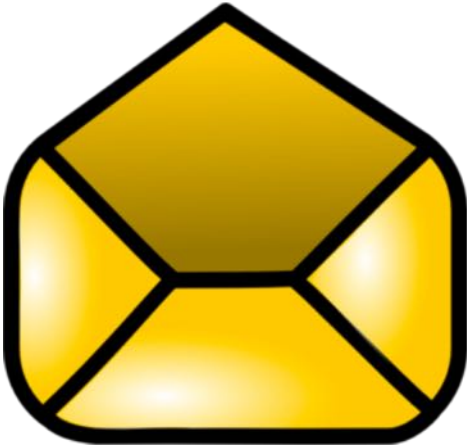
# Requests are like Remote Method Calls



getStatus



# Requests are like Remote Method Calls



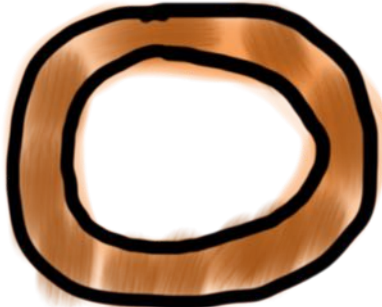
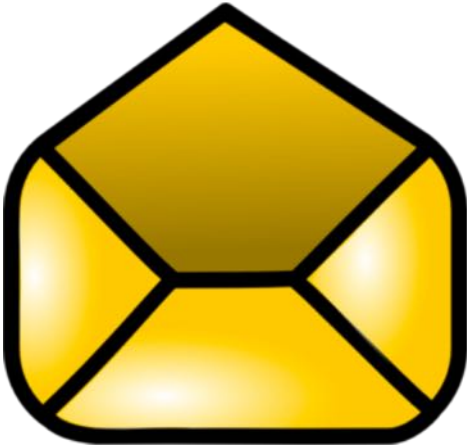
cpiech



getStatus



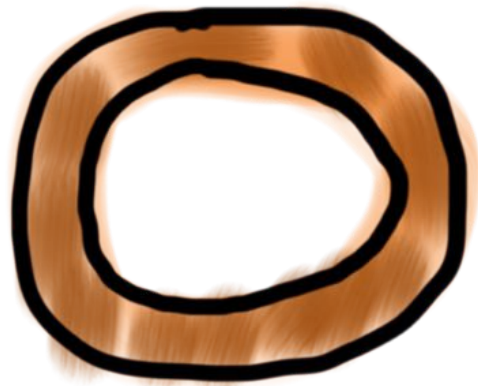
# Requests are like Remote Method Calls



teaching



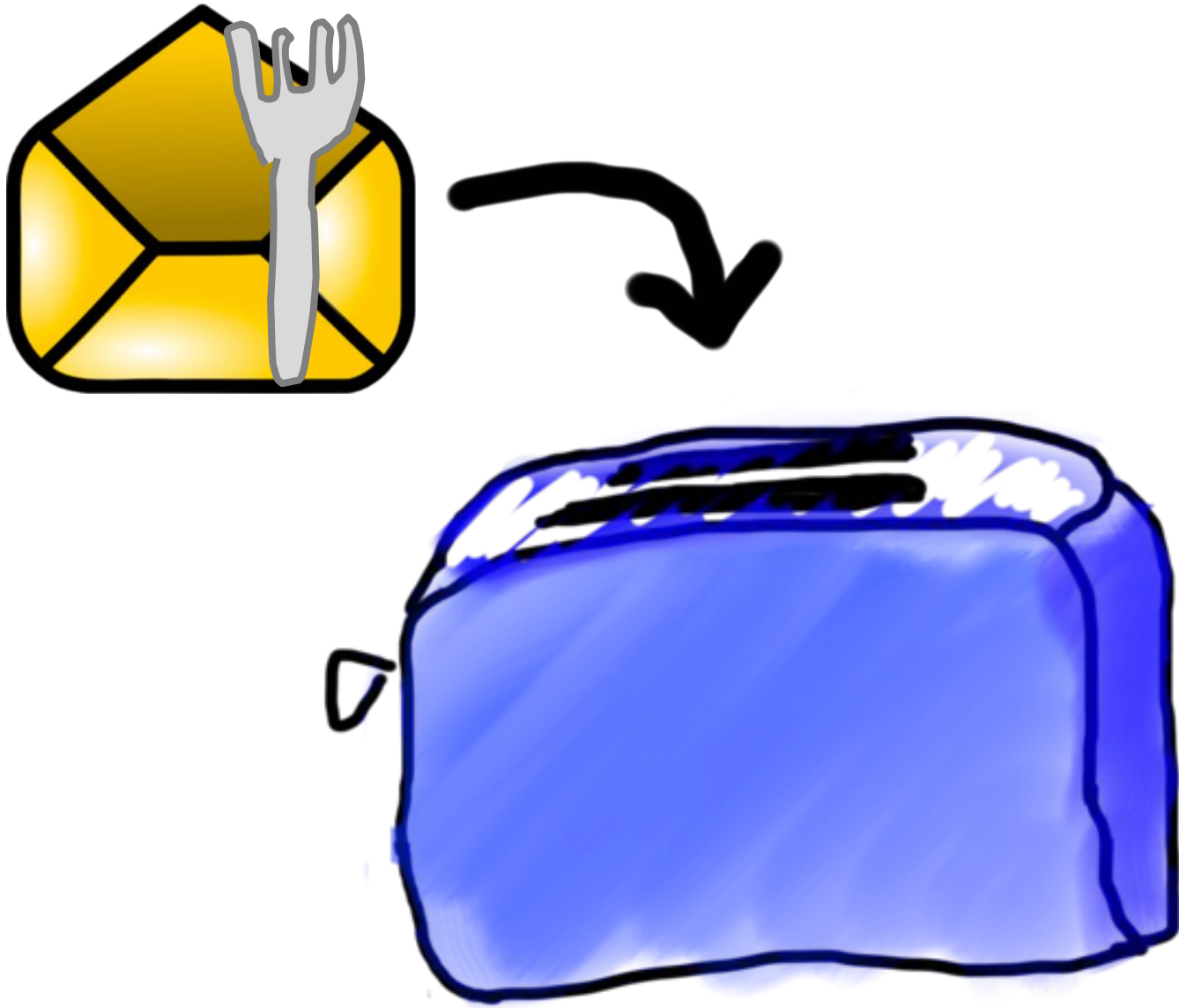
```
public String requestMade(Request request) {  
    String cmd = request.getCommand();  
    if(cmd.equals("getStatus")) {  
        String user = request.getParam("userName");  
        String status = runGetStatus(user);  
        return status;  
    }  
    ...  
}
```



.toString()???



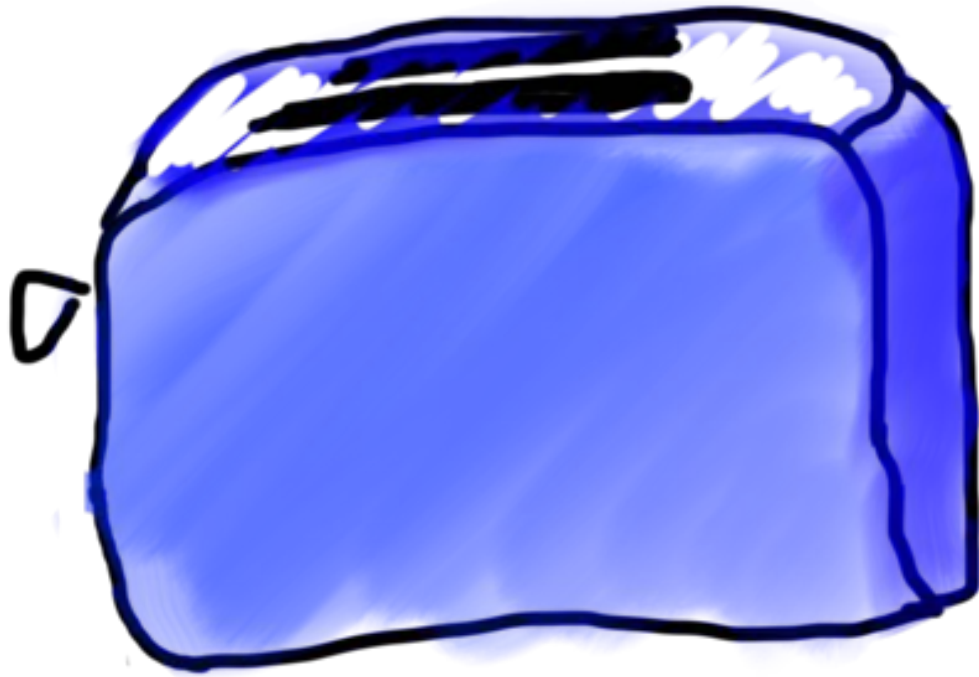
# Requests are like Remote Method Calls



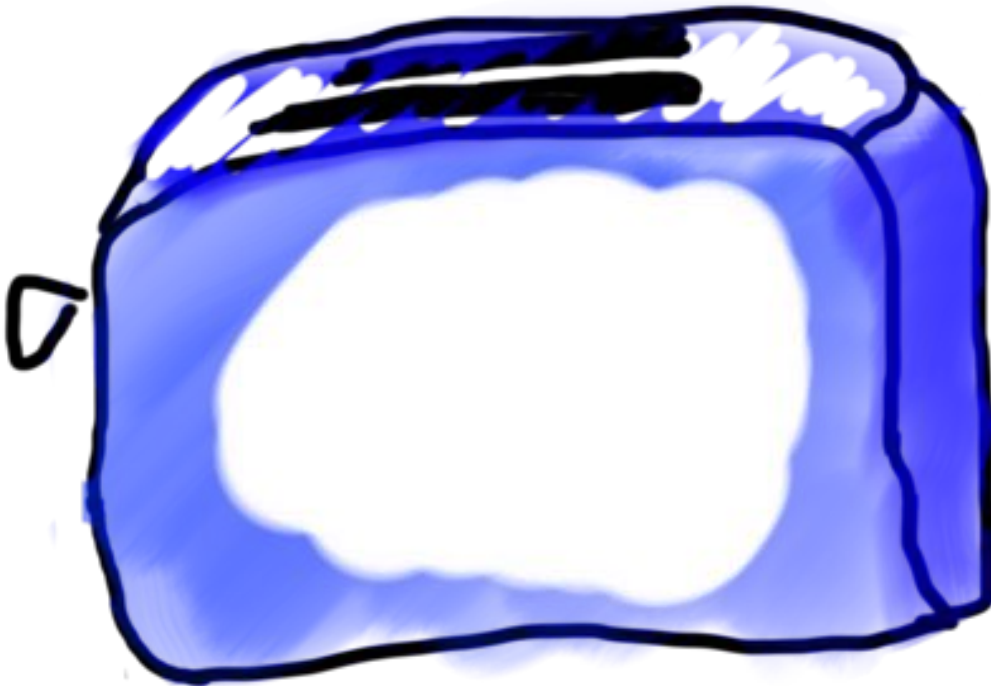
# Requests are like Remote Method Calls



# Requests are like Remote Method Calls



# Requests are like Remote Method Calls

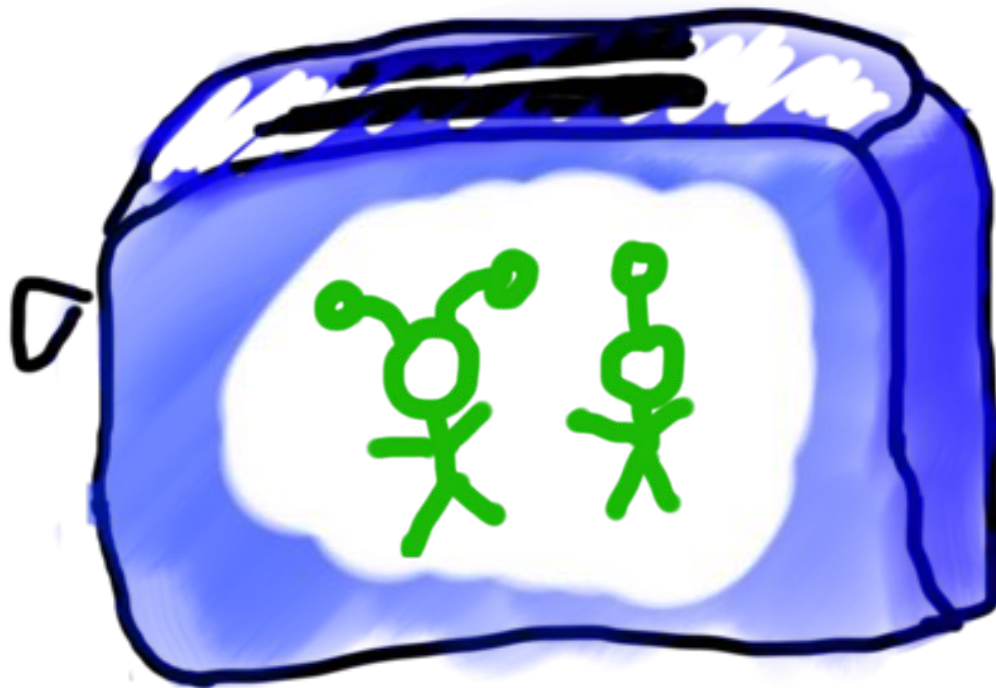




# Requests are like Remote Method Calls



# Requests are like Remote Method Calls



# What is a Request?



```
/* Request has a command */  
String command;
```

```
/* Request has parameters */  
HashMap<String,String> params;
```

Request request

---

```
// methods that the server calls on requests  
request.getCommand();  
request.getParam(key); //returns associated value
```



# A Server's Simple Purpose

1

```
public String requestMade(Request request) {  
    // server code goes here  
}
```

2

```
// make a Server object  
private SimpleServer server  
    = new SimpleServer(this, 8000);
```

3

```
public void run(){  
    // start the server  
    server.start();  
}
```



# A Server's Simple Purpose

1

```
public String requestMade(Request request) {  
    // server code goes here  
}
```

2

```
// make a Server object  
private SimpleServer server  
    = new SimpleServer(this, 8000);
```

3

```
public void run(){  
    // start the server  
    server.start();  
}
```



# What is a Port?



# Servers on one slide

1

```
public String requestMade(Request request) {  
    // server code goes here  
}
```

2

```
// make a Server object  
private SimpleServer server  
    = new SimpleServer(this, 8000);
```

3

```
public void run(){  
    // start the server  
    server.start();  
}
```



# Echo Server





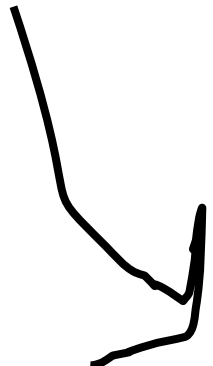
# Echo Server

## Request

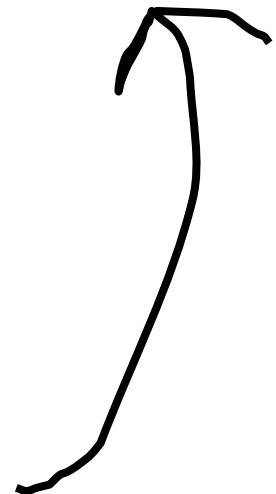
Any Request

## String

Length of the cmd



```
EchoServer
Starting server...
Request recieved hello
Request recieved this+is+a+test
Request recieved whatsGood
Request recieved ping
Request recieved ping
Request recieved ping
Request recieved pong
Request recieved ping
```



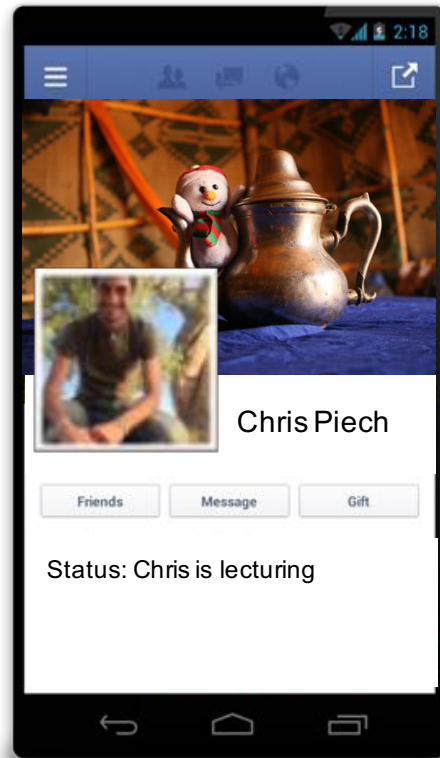


There are two types of  
internet programs. Servers  
and Clients



Then, the client

# A Client's Purpose



1. Interact with the user
2. Get data from its server
3. Save data to its server



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```





# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



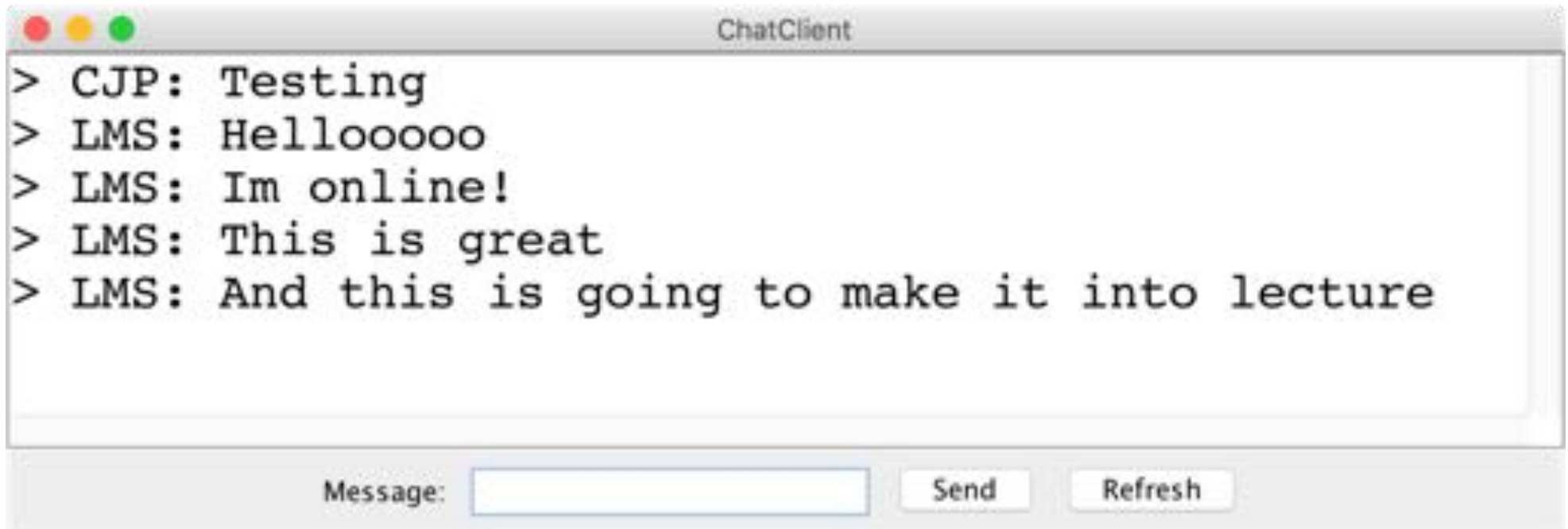
# Clients on one slide

```
try {  
    // 1. construct a new request  
    Request example = new Request("getStatus");  
  
    // 2. add parameters to the request  
    example.addParam("name", "chris");  
  
    // 3. send the request to a computer on the internet  
    String result = SimpleClient.makeRequest(HOST, example);  
} catch(IOException e) {  
    // The internet is a fast and wild world my friend  
}
```



Time for a little chat

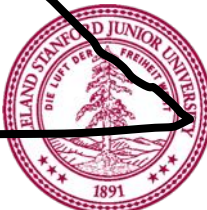
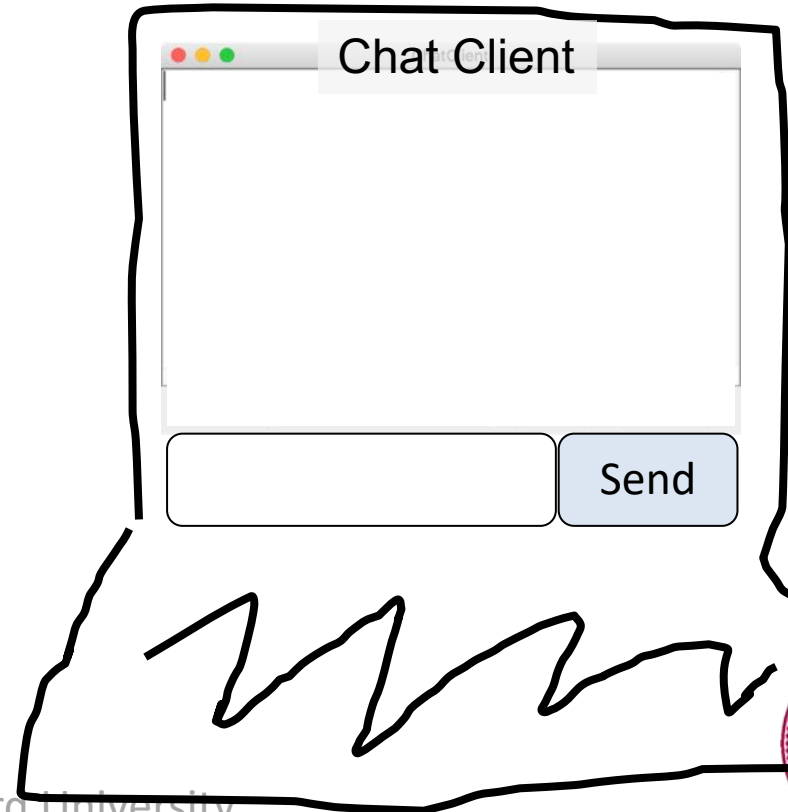
# Chat Server and Client





```
history = [  
]
```

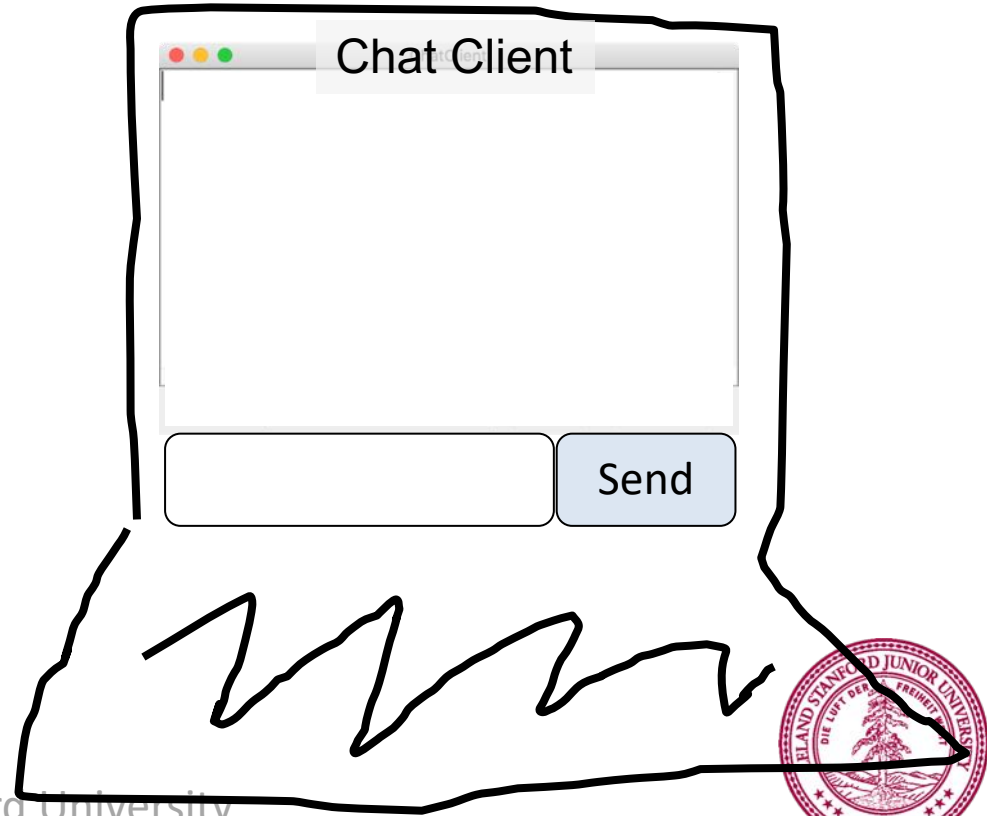
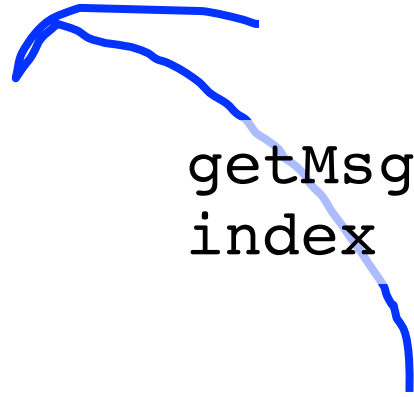
```
addMsg  
msg = C: Hello world
```





```
history = [  
    C: Hello world  
]
```

```
getMsgs  
index = 0
```

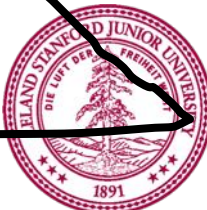
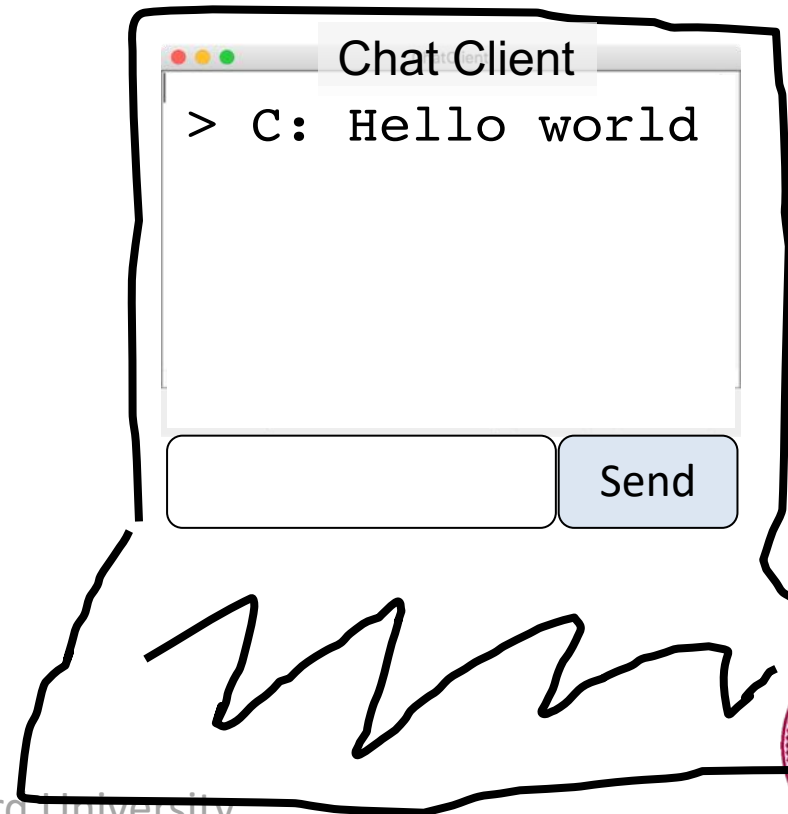
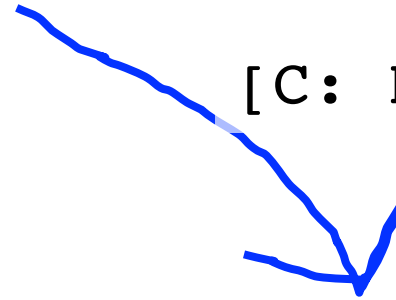






```
history = [  
    C: Hello world  
]
```

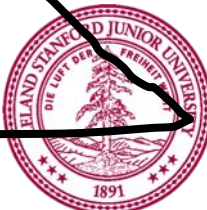
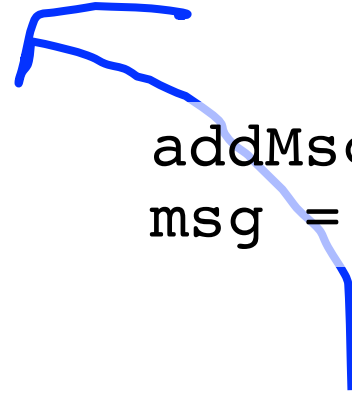
[C: Hello world]





```
history = [  
    C: Hello world  
]
```

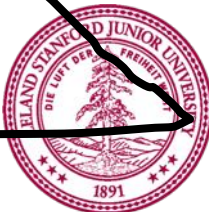
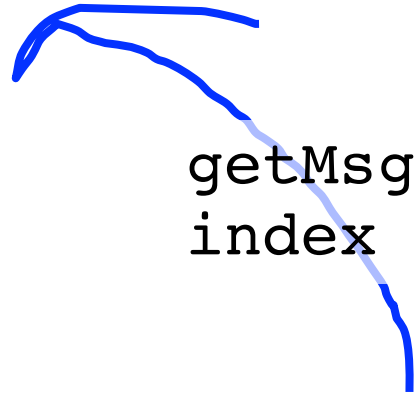
```
addMsg  
msg = B: Im here too
```





```
history = [  
    C: Hello world,  
    B: Im here too  
]
```

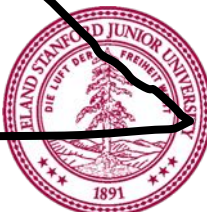
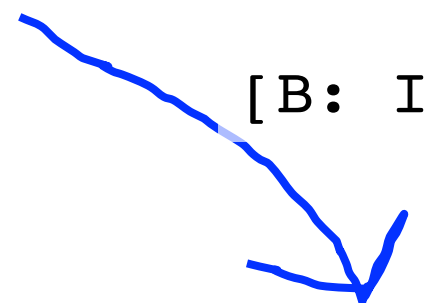
```
getMsgs  
index = 1
```





```
history = [  
  C: Hello world,  
  B: Im here too  
]
```

[B: Im here too]



# Chat Server

Chat Server



```
addMsg  
msg = text
```



```
getMsgs  
index = startIndex
```

