Exercise 1

- Try all the client-server systems discussed in the lecture "Interprocess Communication":

   1. Java based client server system communicating through UDP

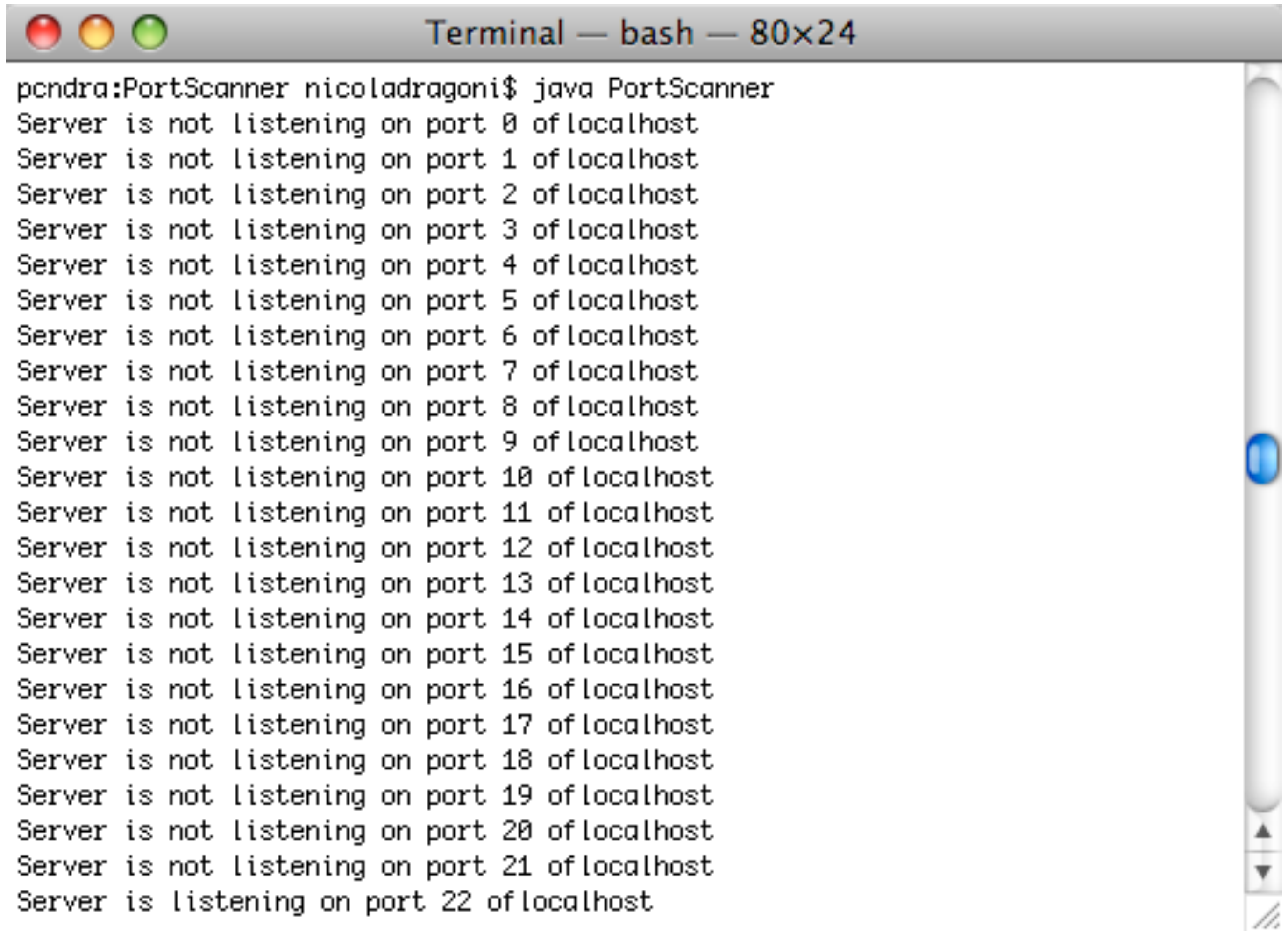
   2. Java based client server system communicating through TCP

Exercise 2

- Implement a client program that
  - repeatedly reads a line of input from the user,
  - sends it to the server in a UDP datagram message,
  - then receives a message from the server.
  - The client sets a timeout on its socket so that it can inform the user when the server does not reply. This can be done with the setSoTimeout() method, for instance: aSocket.setSoTimeout(3000) set the timeout to 3000 milliseconds.

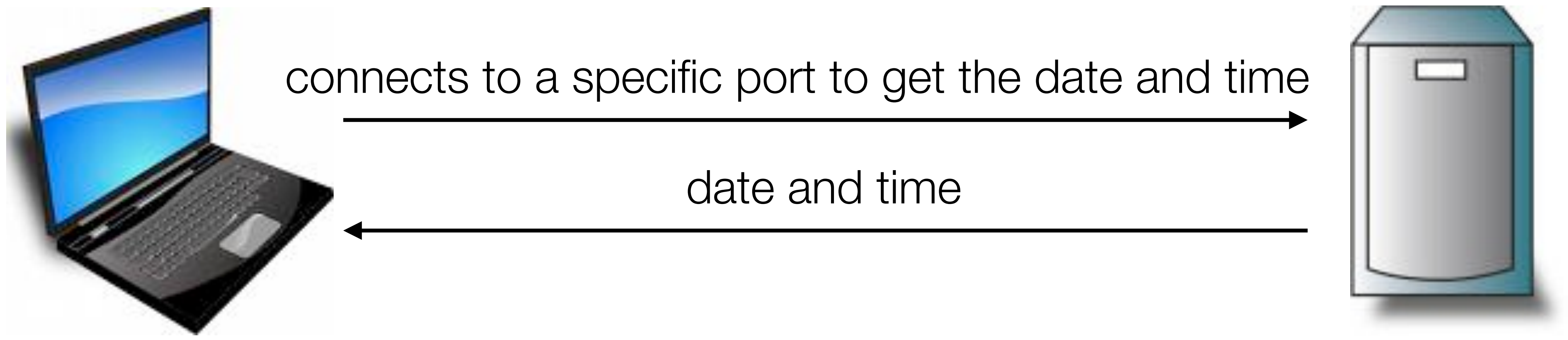
Exercise 3: Port Scanner

- Implement a Java program that acts as a *port scanner*: it checks a number of ports (for instance, from 1 to 1026) to see if they are open (a server is listening on that port number) or closed (a server is not listening on that port number).



```
pcndra:PortScanner nicoladragoni$ java PortScanner
Server is not listening on port 0 of localhost
Server is not listening on port 1 of localhost
Server is not listening on port 2 of localhost
Server is not listening on port 3 of localhost
Server is not listening on port 4 of localhost
Server is not listening on port 5 of localhost
Server is not listening on port 6 of localhost
Server is not listening on port 7 of localhost
Server is not listening on port 8 of localhost
Server is not listening on port 9 of localhost
Server is not listening on port 10 of localhost
Server is not listening on port 11 of localhost
Server is not listening on port 12 of localhost
Server is not listening on port 13 of localhost
Server is not listening on port 14 of localhost
Server is not listening on port 15 of localhost
Server is not listening on port 16 of localhost
Server is not listening on port 17 of localhost
Server is not listening on port 18 of localhost
Server is not listening on port 19 of localhost
Server is not listening on port 20 of localhost
Server is not listening on port 21 of localhost
Server is listening on port 22 of localhost
```

Exercise 4: DayTime Client Server System

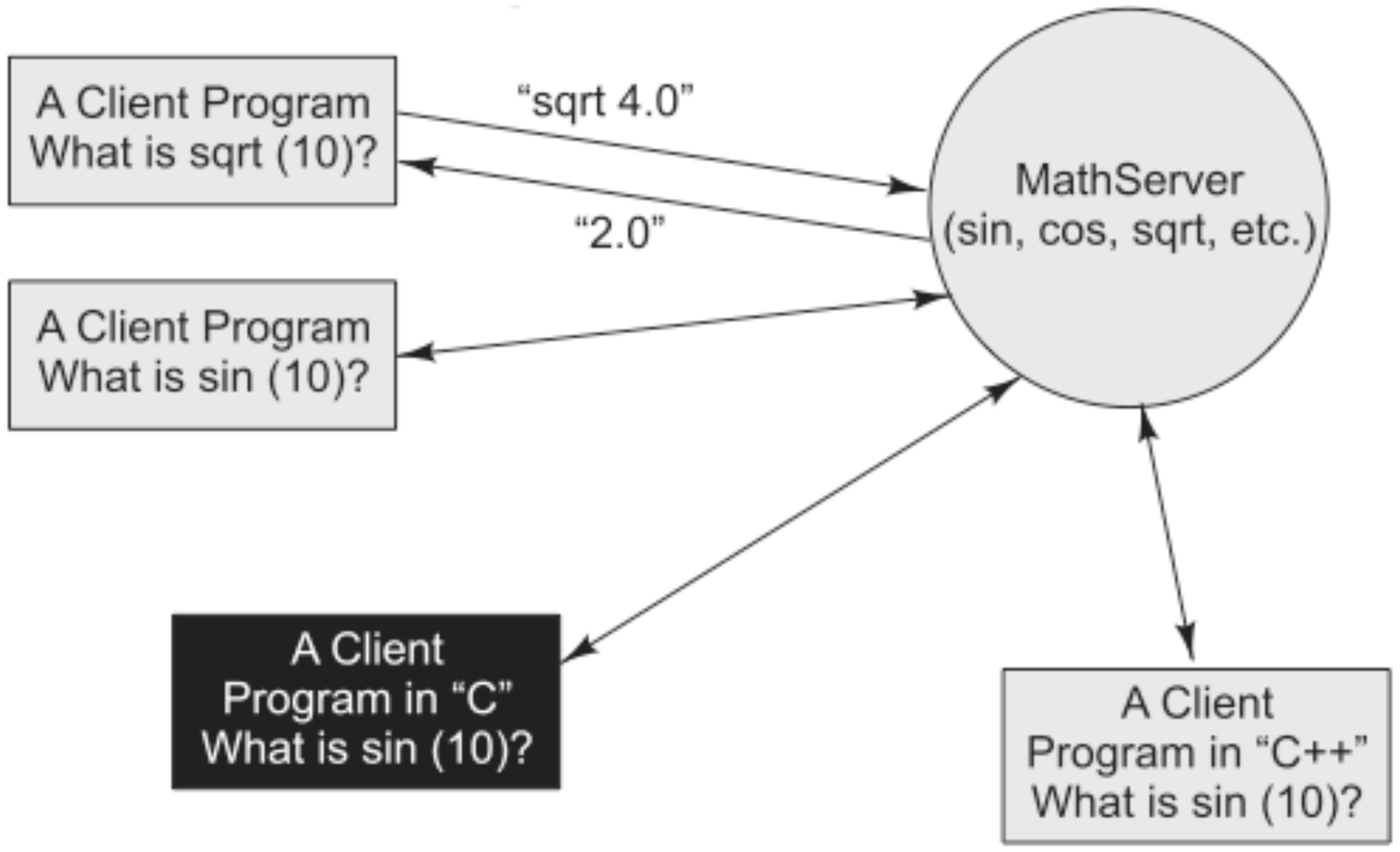


java.util.Date()

Allocates a Date object and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.

Exercise 5: Online Math Server

- Implement a sample math client-server interaction demonstrating online math server that can perform basic math operations.


A Client Program What is sqrt (10)?
"sqrt 4.0"
"2.0"
MathServer (sin, cos, sqrt, etc.)
A Client Program What is sin (10)?
A Client Program in "C" What is sin (10)?
A Client Program in "C++" What is sin (10)?

Exercise 5 (cont.)

- Basic Math interface:

```
// MathService.java: A basic math interface.
public interface MathService {
   public double add(double firstValue, double secondValue);
   public double sub(double firstValue, double secondValue);
   public double div(double firstValue, double secondValue);
   public double mul(double firstValue, double secondValue);
}
```

- N.B.: the implementation of MathServer has to consider the specific protocol used by the math server and the client to communicate.
  - For instance, you can use a very simple protocol operator:

first_value:second_value

It is the math server's responsibility to understand this protocol and delegate to the proper methods such as *add*, *sub*, *mul*, or *div*.