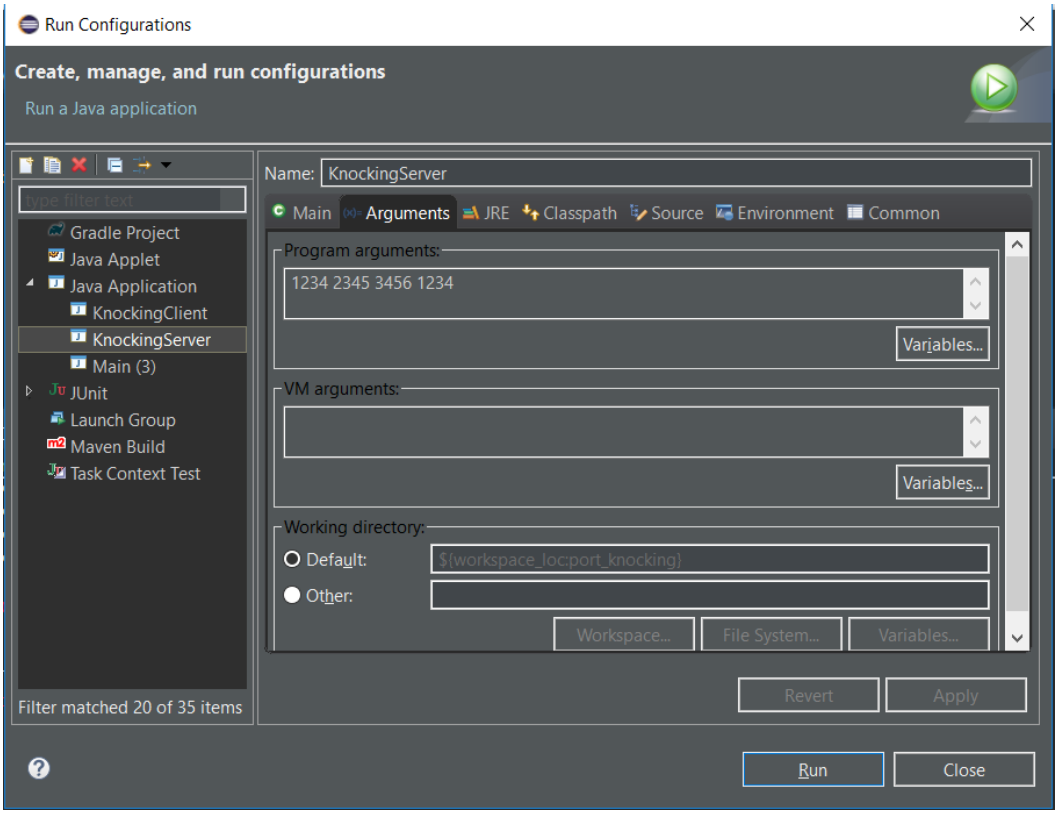
SKJ ASSIGNMENT 2 REPORT

UDP PORT KNOCKING

A. General Solution Description

I. Example of execution

1. Run KnockingServer, passing UDP port sequence as arguments
   For example: 1234 2345 3456 1234



2. Run KnockingClient, passing IP of Server and UDP port sequence as arguments
   For example: localhost 1234 2345 3456 1234
3. See the result
4. Run KnockingClient as many times as needed
5. If UDP port sequence passed to Knocking Client is correct and UDP packets are sent correctly, the result for client side should be (request: a string - response: uppercase all letters of that string):

```
Knocking UDP port: 1234
Knocking UDP port: 2345
Knocking UDP port: 3456
Knocking UDP port: 1234

Successfully knocked ports
Connecting to TCP port 14453
TCP Request: faMC7JPDaeoff
Response from Server: FAMC7JPDAEOFF

Close this client
```

And for server side should be:

```
KEYS: [1234, 2345, 3456, 1234]

Opening UDP port 3456
Opening UDP port 1234
Opening UDP port 2345

Client /127.0.0.1 - 36671 has successfully knocked
Listening on TCP port: 14453
TCP Port accepted the client
Receive a request on TCP Port: faMC7JPDaeoff
Responding: FAMC7JPDAEOFF

Client /127.0.0.1 - 55918 has successfully knocked
Listening on TCP port: 22719
TCP Port accepted the client
Receive a request on TCP Port: PMXECj
Responding: PMXECJ
```

6. Else if the UDP port sequence is incorrect or UDP packets are not properly transferred (some loss happen), the server side displays nothing but the client side should be:

```
Knocking UDP port: 1234
Knocking UDP port: 2345
Knocking UDP port: 3456
Knocking UDP port: 1234

OH NOOOOOOOOOO
Time out -> Terminate
Knocked port sequence is not correct or UDP packet has been lost!

PLEASE TRY THE PROGRAM AGAIN
```

II. Classes

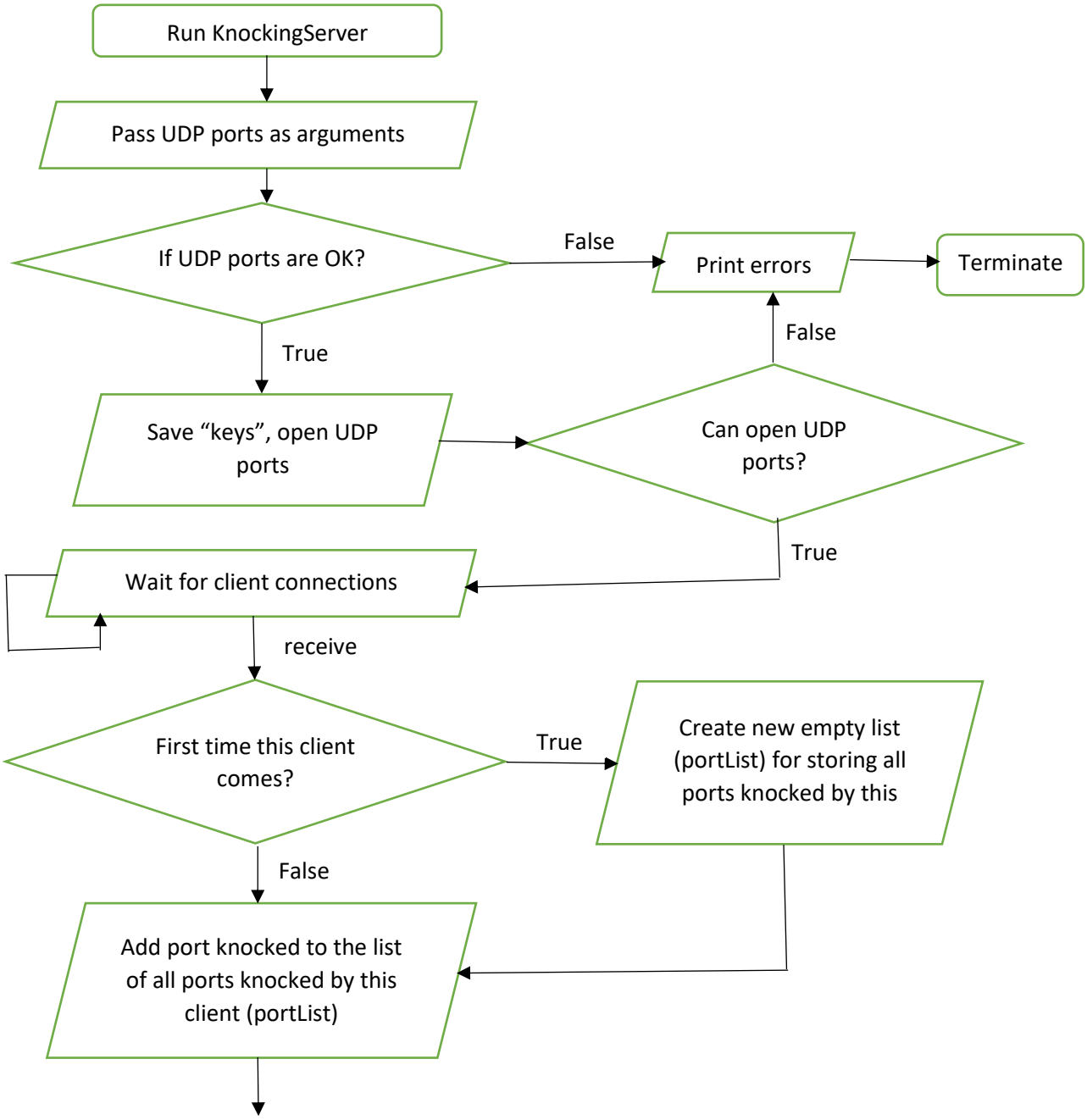
1. Packet server
    a. KnockingServer: Contains static void main method to run the program.
    It checks all arguments, open UDP ports and save sequence of UDP ports in correct order as list “keys”.
    Checking whether a client has knocked correct ports in correct order. If so, open a random TCP port and send an UDP packet to client inform about this TCP port.
    b. ClientInfo: Information about client’s IP and port. (hashCode and equals are overridden).

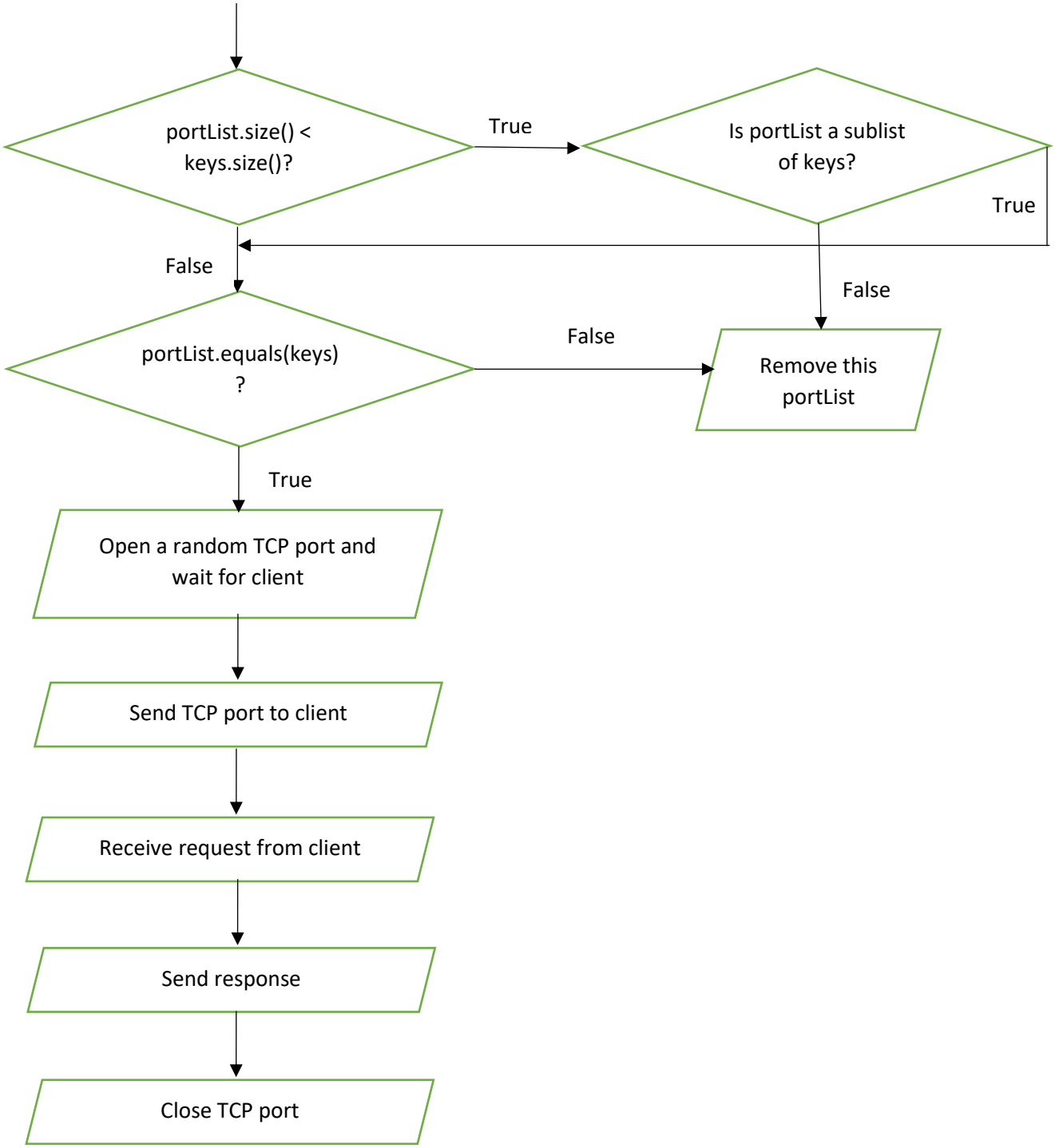
c. UdpReceiveThread (extends Thread): open an UDP ports, listen on them and add new clients or add new ports knocked by clients to the list of newComingClients in class KnockingServer.
d. TcpConnection (extends Thread): open TCP ports, receive requests (each is a string) and send responses (uppercase all letters of that string). And then close the TCP port.

2. Packet client – Class KnockingClient: Check if server IP and all UDP ports are OK, knocking all UDP ports. Set timeout and wait for UDP message from server. Get the TCP port, connect and do a simple request response and then disconnect and terminate. When time is out but client has not receive the UDP message, the program will be terminated.

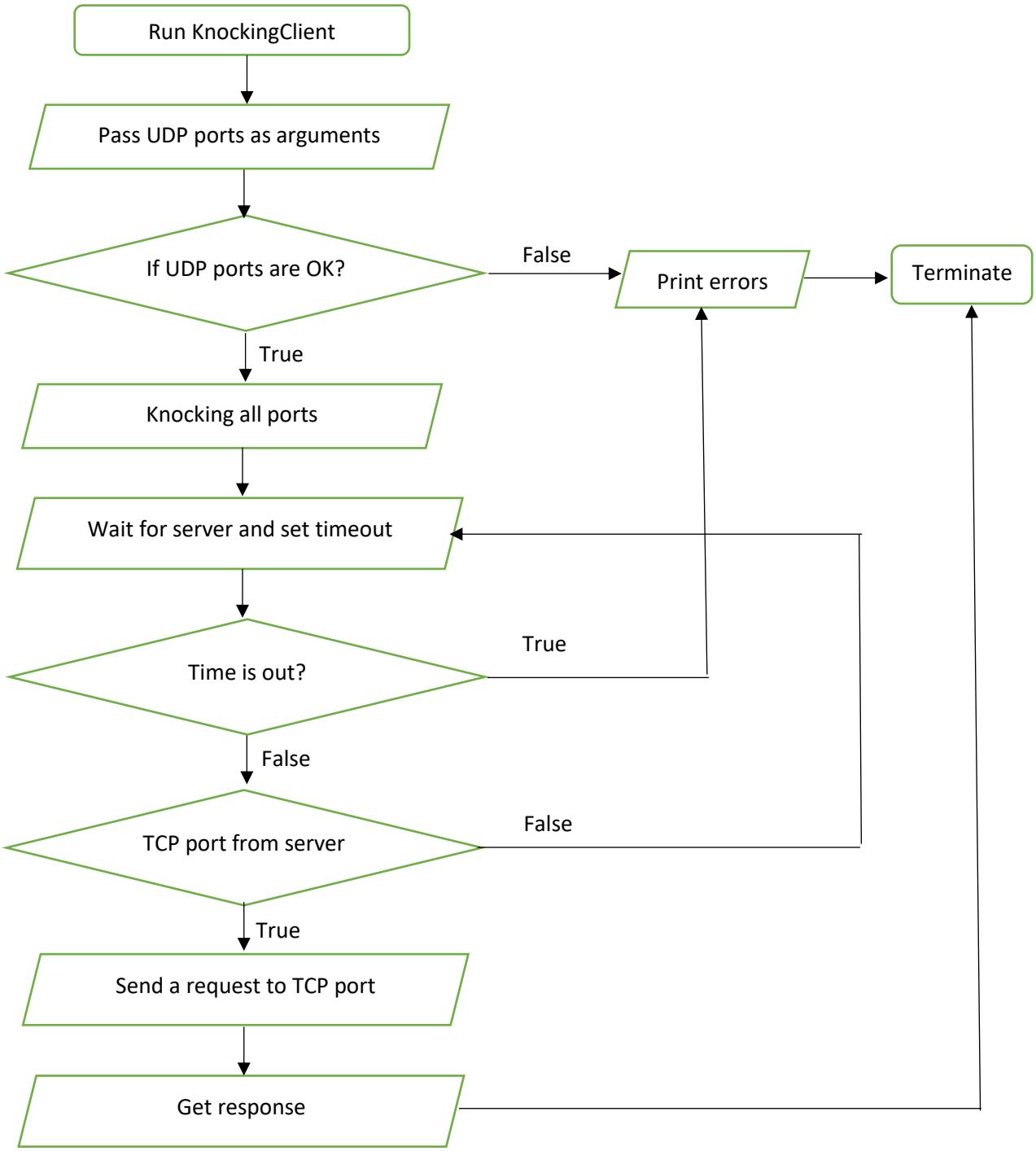
B. Precise solution description

I. Server




portList.size() < keys.size()?
True
Is portList a sublist of keys?
True
False
False
portList.equals(keys) ?
False
Remove this portList
True
Open a random TCP port and wait for client
Send TCP port to client
Receive request from client
Send response
Close TCP port

II. Client



C. Conclusion

The project has been done. The difficulty is opening UDP port and checking the sequence of UDP ports knocked.

Contents