# COM561 - Concurrent and Distributed Computing (2018-19)

### Assignment – Part 2

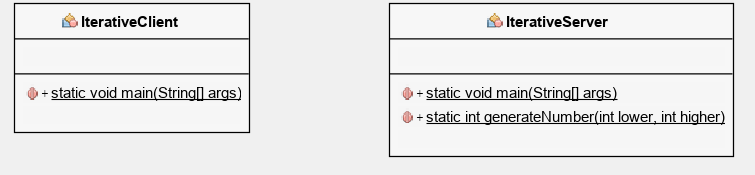
SOLUTIONS

#### Solution 1

1. Single Threaded Client/Server System has the following interaction:

* Server waits for connections from clients
* A client connects to the server and the user inputs the “lowerLimit” and “upperLimit” integer values which are validated on the client side and then sent to the server.
* The server first generates the target number based on the “lowerLimit” and “upperLimit” values and then waits for the guess values to arrive from the client.
* The server reads each guess value and sends back an appropriate response to the client.
* The client displays the response received back and keeps sending (user input) guess values until the target value is matched. Finally, the client displays the number of guesses that were required to match the target value.
* The client closes the communication with the server when a single guessing game is complete.
* However, the server sits waiting for the next client to connect.

**UML Diagram - Client and Iterative Server**



**The client code consists of a single class with a main method that interacts with the server.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class IterativeClient {

public static void main(String[] args) {

String hostName = "LAPTOP-4OJO7OKH"; //Default Host

int portNum = 7000; //Default Port

int guessCounter = 0;

boolean found = false;

Scanner userInput = new Scanner(System.in);

// assign host machine name and port to connect to it

if (args.length != 0) {

if (args[0] != null) {

hostName = args[0]; // user specified machine

}

if (args[1] != null) {

portNum = Integer.parseInt(args[1]); // user specified port

}

}

//Try to connect to server

try(Socket clientSocket = new Socket(hostName, portNum)){

//Try to get input and output streams from server

try(PrintWriter os = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream()));

BufferedReader is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream())))

{

String fromServer;

int userGuess = 0, lowLimit = 0, highLimit = 0;

System.out.print("Welcome to the number guessing game!\n"

+ "You will input two numbers, a lower limit and a higher limit.\n"

+ "I'll think of a number between those two limits and you have to guess it.\n"

+ "GO!");

System.out.println("\nPlease Enter the lower boundary: ");

//Check user input is an integer only

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

lowLimit = userInput.nextInt();

os.println(lowLimit);

os.flush();

System.out.println("\nPlease Enter the higher boundary: ");

//Validation to ensure higher boundary is higher than lower boundary

boolean valid = false;

while(!valid){

//loop to ensure token entered is an integer only

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

//check lower limit isn't greater than higher limit

highLimit = userInput.nextInt();

if(lowLimit >= highLimit){

System.out.println("Lower limit cannot be greater than or equal to higher limit");

}

else{

os.println(highLimit);

os.flush();

valid = true;

}

}

System.out.println("\nGreat! I'll now think of a number...");

//This is where the user starts guessing

while((fromServer = is.readLine()) != null){

System.out.println("Server: " + fromServer);

//if the user finds the selected number

if(fromServer.equals("You got it!")){

System.out.println("\nYou guessed " + guessCounter + " times.");

found = true;

}

//break out of the guessing loop if the number has been guessed corrently

if(found == true){

break;

}

//Check user input is an integer only

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

//take users guess and send it to the server

userGuess = userInput.nextInt();

os.println(userGuess);

os.flush();

guessCounter++; //increment guess after every message sent

}

}catch(IOException e){

System.err.println("Unable to open socket to Host " + e);

}

}catch(IOException e){

System.err.println("Unable to connect to Host " + e);

}

}

}

**The server code also consists of a single class that deals with both the connecting clients and the interactions between clients.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class IterativeServer {

public static void main(String[] args) {

int clientGuesses = 0;

int LOWER\_BOUNDARY = 0;

int HIGHER\_BOUNDARY = 0;

int generatedNumber = 0;

boolean found;

int portNum = 7000; //default port

if(args.length == 1){

portNum = Integer.parseInt(args[1]);

}

System.out.println("Number server started.");

//Creates the server socket that has been chosen

try(ServerSocket serverSocket = new ServerSocket(portNum)){

while(true){

//connect the client

Socket clientSocket = serverSocket.accept();

//When a client connects

try (BufferedReader is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter os = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream())))

{

System.out.println("\nServer Started");

found = false;

String input, output;

//Read set boundaries from user

LOWER\_BOUNDARY = Integer.parseInt(is.readLine());

HIGHER\_BOUNDARY = Integer.parseInt(is.readLine());

//Create number to be guessed

generatedNumber = generateNumber(LOWER\_BOUNDARY, HIGHER\_BOUNDARY);

System.out.println("Lower: " + LOWER\_BOUNDARY);

System.out.println("Higher: " + HIGHER\_BOUNDARY);

System.out.println("Selected: " + generatedNumber);

//inform user number has been generated and to start guessing

output = "Guess a number between " + LOWER\_BOUNDARY + " and " + HIGHER\_BOUNDARY;

os.println(output);

os.flush();

//continue looping while the user keeps sending guesses and found is false

while((input = is.readLine()) != null && found != true){

//if the client's guess is greater than the target

if(Integer.parseInt(input) > generatedNumber){

output = "Lower";

clientGuesses++;

} //if it's lower than the target

else if(Integer.parseInt(input) < generatedNumber){

output = "Higher";

clientGuesses++;

} //otherwise it is the target number

else{

output = "You got it!";

clientGuesses++;

System.out.println("FOUND ");

System.out.println("Client guessed " + clientGuesses + " times");

found = true;

}

os.println(output);

os.flush();

if(found == true){

break;

}

}

}catch(IOException e){

System.out.print("Failed to get I/O Streams" + e);

System.exit(1);

}

}

}

catch(IOException e){

System.out.print("Accept failed on port: " + 4444 + ", " + e);

System.exit(1);

}

}

public static int generateNumber(int lower, int higher){

int num = 0;

Random r = new Random();

num = r.nextInt(higher - lower) + lower;

return num;

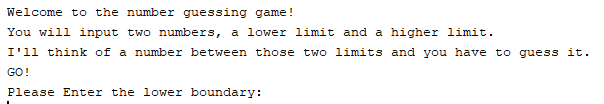
}

}

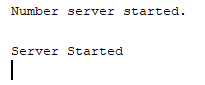
As a result of the server being iterative, only one client can converse with the server at a time. Even though multiple clients can connect to the server at one time, the interactions for client n+1 won’t commence until client n is finished communicating with the server etc.

The typical interactions between client and server are as follows:

**Client:**

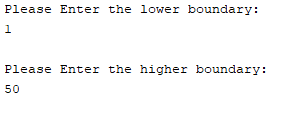


**Server:**

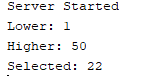


Entering boundaries

**Client:**

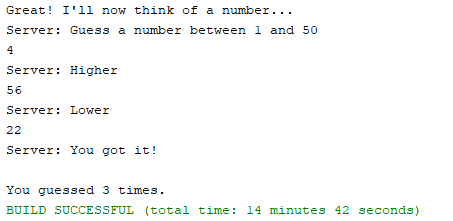


**Server:**

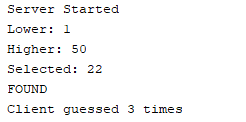


Entering guesses

**Client:**



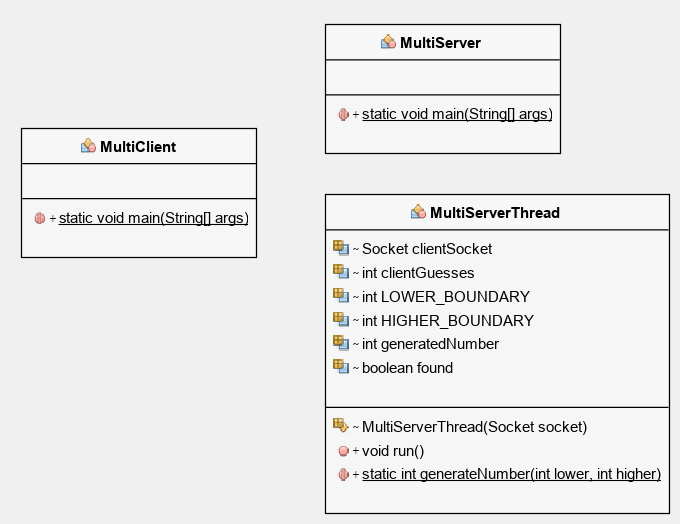
**Server:**



**Solution 2**

(b) Multi-Threaded Server has the exact same interactions as the iterative server does, however it can also communicate with multiple clients concurrently.

**UML Diagram – Client and Concurrent Server**



**The Client code remains completely unchanged and is the exact same.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class MultiClient {

public static void main(String[] args) {

String hostName = "Jack-PC"; //Default Host

int portNum = 7000; //Default Port

int guessCounter = 0;

boolean found = false;

Scanner userInput = new Scanner(System.in);

// assign host machine name and port to connect to it

if (args.length != 0) {

if (args[0] != null) {

hostName = args[0]; // user specified machine

}

if (args[1] != null) {

portNum = Integer.parseInt(args[1]); // user specified port

}

}

//Try to connect to server

try(Socket clientSocket = new Socket(hostName, portNum)){

//Try to get input and output streams from server

try(PrintWriter os = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream()));

BufferedReader is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream())))

{

String fromServer;

int userGuess = 0, lowLimit = 0, highLimit = 0;

try{

//BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Welcome to the number guessing game!\n"

+ "You will input two numbers, a lower limit and a higher limit.\n"

+ "I'll think of a number between those two limits and you have to guess it.\n"

+ "GO!");

System.out.println("\nPlease Enter the lower boundary: ");

//Check user input is an integer only

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

lowLimit = userInput.nextInt();

os.println(lowLimit);

os.flush();

System.out.println("\nPlease Enter the higher boundary: ");

//Validation to ensure higher boundary is higher than lower boundary

boolean valid = false;

while(!valid){

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

highLimit = userInput.nextInt();

if(lowLimit >= highLimit){

System.out.println("Lower limit cannot be greater than or equal to higher limit");

}

else{

os.println(highLimit);

os.flush();

valid = true;

}

}

System.out.println("\nGreat! I'll now think of a number...");

//This is where the user starts guessing

while((fromServer = is.readLine()) != null){

System.out.println("Server: " + fromServer);

//if the user finds the selected number

if(fromServer.equals("You got it!")){

System.out.println("\nYou guessed " + guessCounter + " times.");

found = true;

}

if(found == true){

break;

}

//Check user input is an integer only

while(!userInput.hasNextInt()){

System.out.println("Not a valid number, Try again.");

userInput.next();

}

userGuess = userInput.nextInt();

os.println(userGuess);

os.flush();

guessCounter++; //increment guess after every message sent

}

}catch(IOException e){

System.err.println("Unable to open I/O streams " + e);

}

}catch(IOException e){

System.err.println("Unable to open socket to Host " + e);

}

}catch(IOException e){

System.err.println("Unable to connect to Host " + e);

}

}

}

**Whereas the server is now split into two classes: A server class that handles the connections between clients and the server, and then a server thread class that runs the functionality to the client every time a new instance of the server thread is called.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class MultiServer {

public static void main(String[] args) {

boolean listening = true;

int portNum = 7000; //default port

if(args.length == 1){

portNum = Integer.parseInt(args[1]);

}

//Creates the server socket that has been chosen

try(ServerSocket serverSocket = new ServerSocket(portNum)){

System.out.println("Number server started and listening...");

while(listening){

Socket clientSocket;

try{

clientSocket = serverSocket.accept();

}catch(IOException e){

System.err.println("Accept failed: " + portNum + ", " + e.getMessage());

continue;

}

new MultiServerThread(clientSocket).start();

}

}

catch(IOException e){

System.out.print("Couldn't listen on port: " + portNum + ", " + e);

System.exit(1);

}

}

}

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class MultiServerThread extends Thread {

Socket clientSocket = null;

int clientGuesses = 0;

int LOWER\_BOUNDARY = 0;

int HIGHER\_BOUNDARY = 0;

int generatedNumber = 0;

boolean found;

MultiServerThread(Socket socket){

super("MultServerThread");

this.clientSocket = socket;

}

@Override

public void run(){

//When a client connects

try (BufferedReader is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter os = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream())))

{

System.out.println("New instance of game starting");

found = false;

String input, output;

//Read set boundaries from user

LOWER\_BOUNDARY = Integer.parseInt(is.readLine());

HIGHER\_BOUNDARY = Integer.parseInt(is.readLine());

//Create number to be guessed

generatedNumber = generateNumber(LOWER\_BOUNDARY, HIGHER\_BOUNDARY);

System.out.println("Lower: " + LOWER\_BOUNDARY);

System.out.println("Higher: " + HIGHER\_BOUNDARY);

System.out.println("Selected: " + generatedNumber);

output = "Guess a number between " + LOWER\_BOUNDARY + " and " + HIGHER\_BOUNDARY;

os.println(output);

os.flush();

while((input = is.readLine()) != null && found != true){

if(Integer.parseInt(input) > generatedNumber){

output = "Lower";

clientGuesses++;

}

else if(Integer.parseInt(input) < generatedNumber){

output = "Higher";

clientGuesses++;

}

else{

output = "You got it!";

clientGuesses++;

System.out.println("FOUND ");

System.out.println("Client guessed " + clientGuesses + " times");

found = true;

}

os.println(output);

os.flush();

if(found == true){

break;

}

}

}

catch(IOException e){

System.out.print("Failed to get I/O Streams" + e);

System.exit(1);

}

System.out.println("Client left");

}

public static int generateNumber(int lower, int higher){

int num = 0;

Random r = new Random();

num = r.nextInt(higher - lower) + lower;

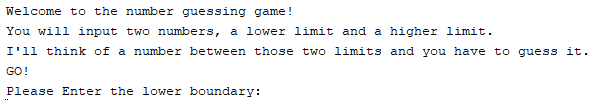
return num;

}

}

Typical interactions between the server and multiple clients connected at one time are outlined as follows:

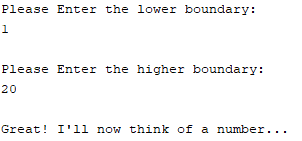
**Client 1:**



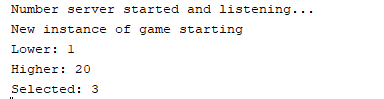
**Server:**



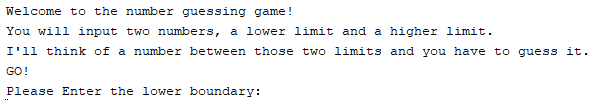
**Client 1 inputting boundaries:**



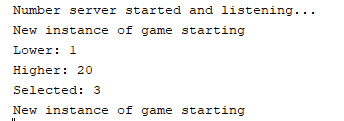
**Server:**



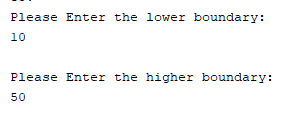
**Client 2 Connects**



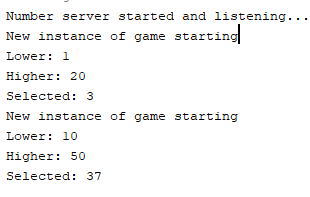
**Server:**



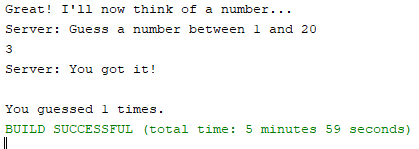
**Client 2 inputting boundaries:**



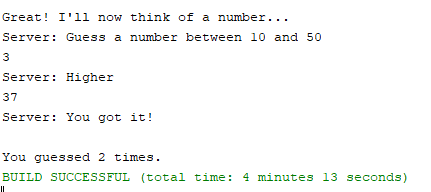
**Server:**



**Client 1 guessing:**



**Client 2 guessing:**



**Server:**

