**Project2Task4Server**

import java.net.\*;  
import java.io.\*;  
import java.util.Map;  
import java.util.Scanner;  
import java.util.TreeMap;  
// Name: Leo Lin  
// Andrew ID: hungfanl  
public class RemoteVariableServerTCP {  
 static Map<Integer, Integer> *database* = new TreeMap<>();  
 public static void main(String args[]) {  
 Socket clientSocket = null;  
 try {  
 int serverPort = 7777; // the server port we are using  
 // Create a new server socket  
 ServerSocket listenSocket = new ServerSocket(serverPort);  
 clientSocket = listenSocket.accept();  
 // If we get here, then we are now connected to a client.  
 // Set up "in" to read from the client socket  
 Scanner in;  
 in = new Scanner(clientSocket.getInputStream());  
 // Set up "out" to write to the client socket  
 PrintWriter out;  
 out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream())));  
 // In order for the server to run forever, we need to handle the situation when client shut down  
 while (true) {  
 String data;  
 // When the client is connected, there will be string pass to in.  
 // Under the situation, we can safely conduct the calculation and return the value.  
 if(in.hasNextLine()){  
 data = in.nextLine();  
 String[] operation = data.split(",");  
 int operator = Integer.*parseInt*(operation[0]);  
 int id = Integer.*parseInt*(operation[1]);  
 int number = Integer.*parseInt*(operation[2]);  
 // conduct corresponding arithmetic according to the three index  
 int outcome = *arithmetic*(operator, id, number);  
 // return the outcome of calculation to the client  
 System.*out*.println("Returning sum of " + outcome + " to client");  
 out.println(outcome);  
 out.flush();  
 }  
 // However, when the client is shut down, there will not be next line pass to in.  
 // In such cases, we will need to have clientSocket to accept another socket and renew the in/out  
 else {  
 clientSocket = listenSocket.accept();  
 in = new Scanner(clientSocket.getInputStream());  
 out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream())));  
 }  
 }  
 // Handle exceptions  
 } catch (IOException e) {  
 System.*out*.println("IO Exception:" + e.getMessage());  
  
 // If quitting (typically by you sending quit signal) clean up sockets  
 } finally {  
 try {  
 if (clientSocket != null) {  
 clientSocket.close();  
 }  
 } catch (IOException e) {  
 // ignore exception on close  
 }  
 }  
 }  
 // Doing corresponding arithmetic according to the user's request  
 public static int arithmetic(int operator, int id, int number){  
 String[] operation = {"Addition", "Subtraction", "Check"};  
 System.*out*.println("The visitor's ID is: " + id);  
 System.*out*.println("Operator: " + operation[operator-1]);  
 if(operator == 1){  
 *database*.put(id, *database*.getOrDefault(id,0) + number);  
 }  
 else if (operator == 2){  
 *database*.put(id, *database*.getOrDefault(id,0) - number);  
 }  
 // return the number after calculation  
 return *database*.get(id);  
 }  
}