import java.io.\*;

import java.net.\*;

import java.util.\*;

class Worker extends Thread {

    Socket sock;

    private static HashMap<UUID, List<Integer>> uuidJokeListHashMap = new HashMap<>();

    private static HashMap<UUID, List<Integer>> uuidProverbListHashMap = new HashMap<>();

    private static HashMap<UUID, String> uuidNameHashMap = new HashMap<>();

    // private static List<Integer> jokeIndexOrderList = null;

    // private static List<Integer> proverbIndexOrderList = null;

    public static final String[] jokeList = { "JA", "JB", "JC", "JD" };

    public static final String[] proverbList = { "PA", "PB", "PC", "PD" };

    private static void initJokeProverbIndexOrderList(List<Integer> list) {

        list.clear();

        list.addAll(Arrays.asList(0, 1, 2, 3));

        Collections.shuffle(list);

        System.out.println(list.toString());

    }

    Worker(Socket s) {

        sock = s;

        // if (jokeIndexOrderList == null) {

        //  jokeIndexOrderList = new ArrayList<>();

        //  initJokeProverbIndexOrderList(jokeIndexOrderList);

        // }

        // if (proverbIndexOrderList == null) {

        //  proverbIndexOrderList = new ArrayList<>();

        //  initJokeProverbIndexOrderList(proverbIndexOrderList);

        // }

    }

    public void run() {

        PrintStream out = null;

        BufferedReader in = null;

        try {

            in = new BufferedReader(new InputStreamReader(sock.getInputStream()));

            out = new PrintStream(sock.getOutputStream());

            String clientMessage = in.readLine();

            UUID uuid;

            List<Integer> list;

            try {

                // if client send uuid, send joke/proverb back

                uuid = UUID.fromString(clientMessage);

                System.out.print("client: " + uuid.toString() + " ");

                switch (JokeServer.mode) {

                    case JOKE\_MODE:

                         list = uuidJokeListHashMap.get(uuid);

                        sendJokeToClient(out, list);

                        break;

                    case PROVERB\_MODE:

                        list = uuidProverbListHashMap.get(uuid);

                        sendProverbToClient(out, list);

                        break;

                }

            } catch (IllegalArgumentException iae) { // client not send uuid, assume is a name

                uuid = UUID.randomUUID();

                String name = clientMessage;

                uuidNameHashMap.put(uuid, name);

                list = new ArrayList<>();

                initJokeProverbIndexOrderList(list);

                uuidJokeListHashMap.put(uuid, list);

                list = new ArrayList<>();

                initJokeProverbIndexOrderList(list);

                uuidProverbListHashMap.put(uuid, list);

                // send uuid back to client

                out.println(uuid.toString());

                out.flush();

            }

            sock.close();

        } catch (IOException ioe) {

            System.out.println("server: read error");

            ioe.printStackTrace();

        }

    }

    private void sendJokeToClient(PrintStream out, List<Integer> list) {

        int index;

        String joke;

        if (list.size() > 1) {

            index = list.get(0);

            list.remove(0);

            joke = jokeList[index];

            out.println(joke);

            System.out.println("server: send to client: " + joke);

        } else if (list.size() == 1) { // last joke, size = 1

            index = list.get(0);

            list.remove(0);

            joke = jokeList[index];

            System.out.println("server: send to client: " + joke);

            System.out.println("server: Joke Cycle Completed");

            out.println("Joke Cycle Completed"); // send Dompleted

            out.println(joke); // send message to client

            // reset textJokeProverb order

            initJokeProverbIndexOrderList(list);

        }

    }

    private void sendProverbToClient(PrintStream out, List<Integer> list) {

        int index;

        String proverb;

        if (list.size() > 1) {

            index = list.get(0);

            list.remove(0);

            proverb = proverbList[index];

            out.println(proverb);

            System.out.println("server: send to client: " + proverb);

        } else if (list.size() == 1) { // last proverb, size = 1

            index = list.get(0);

            list.remove(0);

            proverb = proverbList[index];

            System.out.println("server: send to client: " + proverb);

            System.out.println("server: Proverb Cycle Completed");

            out.println("Proverb Cycle Completed"); // send Completed

            out.println(proverb); // send message to client

            // reset textproverbProverb order

            initJokeProverbIndexOrderList(list);

        }

    }

}

class AdminWorker extends Thread {

    Socket sock;

    AdminWorker(Socket s) {

        sock = s;

    }

    public void run() {

        PrintStream out = null;

        BufferedReader in = null;

        try {

            in = new BufferedReader(new InputStreamReader(sock.getInputStream()));

            out = new PrintStream(sock.getOutputStream());

            switch (JokeServer.mode) {

                case JOKE\_MODE:

                    JokeServer.mode = JokeServer.JokeMode.PROVERB\_MODE;

                    out.println("Proverb Mode");

                    out.flush();

                    System.out.println("server: change mode: Proverb");

                    break;

                case PROVERB\_MODE:

                    JokeServer.mode = JokeServer.JokeMode.JOKE\_MODE;

                    out.println("Joke Mode");

                    out.flush();

                    System.out.println("server: change mode: Joke");

                    break;

                default:

                    System.out.println("server: unknown mode:");

                    break;

            }

            sock.close();

        } catch (IOException ioe) {

            System.out.println("server: read error");

            ioe.printStackTrace();

        }

    }

}

public class JokeServer {

    public static JokeMode mode = JokeMode.JOKE\_MODE;

    enum JokeMode {

        JOKE\_MODE,

        PROVERB\_MODE

    }

    public static void main(String a[]) throws IOException {

        JokeAdminServer jas = new JokeAdminServer();

        Thread adminThread = new Thread(jas);

        adminThread.start();

        int q\_len = 6;

        int port = 1565;

        Socket sock;

        ServerSocket servsock = new ServerSocket(port, q\_len);

        System.out.println("Sornthorn Anujavanit's JokeServer server 1.8.4 starting up, listening at port 1565.\n");

        while (true) {

            sock = servsock.accept();

            new Worker(sock).start();

        }

    }

}

class JokeAdminServer implements Runnable {

    JokeAdminServer() {

        System.out

                .println("Sornthorn Anujavanit's JokeAdminServer server 1.8.4 starting up, listening at port 44444.\n");

    }

    public void run() {

        int q\_len = 6;

        int port = 44444;

        Socket sock;

        try {

            ServerSocket servsock = new ServerSocket(port, q\_len);

            while (true) {

                sock = servsock.accept();

                new AdminWorker(sock).start();

            }

        } catch (IOException ioe) {

            System.out.println("server: read error");

            ioe.printStackTrace();

        }

    }

}

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class JokeClient {

    public static String name = null;

    public static String uuidString = null;

    public static void main(String args[]) {

        String serverName;

        if (args.length < 1)

            serverName = "localhost";

        else

            serverName = args[0];

        System.out.println("Sornthorn Anujavanit's JokeClient, 1.8.3\n");

        System.out.println("Using server: " + serverName + ", Port: 1565");

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

        try {

            String input;

            do {

                System.out.print("Enter name, or (quit) to end: ");

                input = in.readLine();

                if (! input.trim().isEmpty() ) {

                    name = input;

                }

            } while (name ==  null) ;

            //

            while (uuidString == null) {

                // send name to server, wait for uuid

                // input = in.readLine();

                Socket sock = new Socket(serverName, 1565);

                PrintStream toServer = new PrintStream(sock.getOutputStream());

                toServer.println(name);

                BufferedReader fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream()));

                uuidString = fromServer.readLine();

                System.out.println("client: uuid: " + uuidString);

            }

            do {

                // System.out.print("Enter a name, or (quit) to end: ");

                System.out.print("Press (Enter), or (quit) to end: ");

                input = in.readLine();

                Socket sock = new Socket(serverName, 1565);

                PrintStream toServer = new PrintStream(sock.getOutputStream());

                toServer.println(uuidString); // send uuid to server, get joke/proverb

                BufferedReader fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream()));

                String serverMessage;

                serverMessage = fromServer.readLine();

                switch (serverMessage) {

                    case "Joke Cycle Completed":

                        serverMessage = fromServer.readLine(); // read joke

                        System.out.println("client: read from server: " + serverMessage);

                        System.out.println("Joke Cycle Completed" + System.lineSeparator());

                        break;

                    case "Proverb Cycle Completed":

                        serverMessage = fromServer.readLine(); // read proverb

                        System.out.println("client: read from server: " + serverMessage);

                        System.out.println("Proverb Cycle Completed" + System.lineSeparator());

                        break;

                    default:

                        System.out.println("client: read from server: " + serverMessage);

                        break;

                }

                sock.close();

            } while (!input.equalsIgnoreCase("quit"));

            System.out.println("Cancelled by user request.");

        } catch (IOException ioe) {

            ioe.printStackTrace();

        }

    }

}

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class JokeAdminClient {

    public static void main(String args[]) {

        String serverName;

        serverName = "localhost";

        System.out.println("Sornthorn Anujavanit's JokeAdminClient, 1.8.\n");

        System.out.println("Using server: " + serverName + ", Port: 44444");

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

        try {

            String input;

            do {

                System.out.print("Press <Enter> to change mode,  or (quit) to end: ");

                input = in.readLine();

                Socket sock = new Socket(serverName, 44444);

                BufferedReader fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream()));

                String serverMessage;

                try {

                    serverMessage = fromServer.readLine();

                    System.out.println("client: change mode to: " + serverMessage);

                } catch (Exception ex) {

                    System.out.println("error");

                    ex.printStackTrace();

                }

                sock.close();

            } while (!input.equalsIgnoreCase("quit"));

            System.out.println("Cancelled by user request.");

        } catch (IOException ioe) {

            System.out.println("server: read error");

            ioe.printStackTrace();

        }

    }

}