Segona pràctica obligatòria

Xarxes — Curs 2020-2021

Grau en Tècniques d'Interacció Digital i de Computació

Base de dades de personatges en línia

L'associació de jugadors de rol de Sant Esteve de les Roures té un ordinador amb una petita base de dades de personatges. Qualsevol membre de l'associació podia accedir-hi per consultar, afegir o eliminar personatges.

A causa d'un període de confinament, s'han plantejat que estaria bé que els membres poguessin accedir a la base de dades des de casa. El software original, que data del 2019, va ser programat en Java. Així, l'objectiu de la pràctica consisteix en adaptar el software per tal que funcioni en remot, amb un model client-servidor.

La versió local

En la carpeta "Pràctica 2" de la secció de recursos de l'assignatura del campus virtual trobareu una versió local del software plenament funcional. Aquesta està basada en una pràctica de l'assignatura de Programació Orientada a Objectes que molts de vosaltres recordareu. Però no és exactament el mateix codi. Els canvis principals són que s'ha simplificat la informació dels personatges, la re-generació de la base de dades d'exemple és un programa separat i s'ha afegit l'opció d'inserir nous personatges.

Tot i que la informació dels personatges s'emmagatzema en el fitxer binari `charactersDB.dat`, hi ha una carpeta (anomenada `Characters`) que conté uns quants fitxers de text amb informació de personatges per tal que sigui possible regenerar la base de dades d'exemple còmodament.

Per regenerar la base de dades només cal que executeu la classe `GenerateCharactersDB`. Aquesta 'reseteja' la base de dades. Tot seguit, llegeix el fitxer `characters.txt`, que conté el llistat dels fitxers de la carpeta `Characters`, i després procedeix a llegir cadascun dels fitxers per afegir els personatges a la base de dades.

La classe `Main`, que també és executable, es limita a fer servir la base de dades tal com la troba. Si el fitxer `charactersDB.dat` encara no existeix, el crearà buit.

La vostra feina consisteix en adaptar el programa `Main` per tal que funcioni en mode client servidor. Així, haureu de separar la classe `Main.java` en dos (`Server.java` i `Client.java`). El programa `GenerateCharactersDB` se suposa que l'anirà a executar un membre de l'associació de forma local. Per tant, aquest no cal que l'adapteu.

Classes del projecte

Ara mateix el projecte consta de sis classes:

- `CharacterInfo`: conté la informació d'un personatge.
- `CharacterInfoReader`: només té el mètode `readCharacterFile` que llegeix el fitxer d'un personatge.
- `CharactersDB`: conté els mètodes propis de la base de dades.
- `PackUtils`: eïnes d'empaquetat i desempaquetat de dades.
- `GenerateCharactersDB`: programa que regenera la base de dades.
- `Main`: programa principal amb les diverses opcions esmenades.

Vosaltres substituireu la classe `Main` per les dues classes:

- `Server`: espera ordres del `Client` i accedeix a la base de dades per complir-les.
- `Client`: mostra el menú d'opcions i es connecta al `Server` per a que les executi.

Informació d'un personatge

Tal com podeu veure a la classe `CharacterInfo`, la informació d'un personatge és:

- `name`: el nom, un `String` limitat a 26 caràcters.
- `intelligence`: la intel·ligència, que és un `int`.
- `strength`: la força, que és un `int`.
- `constitution`: la constitució, que és un `int`.

Algunes de les funcionalitats del programa requeriran que envieu (o rebeu) la informació d'un personatge per la xarxa. Per tal de fer-ho, podeu fer servir els mètodes `toBytes()` i `fromBytes(byte[])` de la classe `CharacterInfo` que converteixen la informació del personatge en un buffer de bytes, i a la inversa, i llavors enviar-ho amb el mètode `write(byte[])` o rebre-ho amb `read(byte[])`, de les classes `DataOutputStream` i `DataInputStream`, respectivament.

Adaptació del programa

Si executeu el programa `Main` tal com està veureu que us dóna una sèrie d'opcions:

```
_____________________________________
Menú d'opcions:
1 - Llista tots els noms de personatge.
2 - Obté la informació d'un personatge.
3 - Afegeix un personatge.
4 - Elimina un personatge.
5 - Sortir.
Escull una opció:
_____________________________________
```

La vostra tasca és fer que sigui el `Client` qui mostri les opcions i el `Server` qui les executi. A l'hora d'adaptar el programa a un model client-servidor teniu dues opcions:

a) Si feu un servidor interactiu, llavors el client hauria d'obrir la connexió just abans d'enviar cada petició al servidor, i tancar-la tan bon punt arribi la resposta (per tal de tindre el servidor ocupat el menor temps possible). El client ha de tractar l'error de servidor ocupat. Podeu donar un missatge (e.g. "`Servidor ocupat. Torna-ho a intentar aviat.`") o que el mateix programa reintenti connectar-se cada poc temps.

b) Si feu un servidor concurrent, haureu d'assegurar-vos que l'accés a la base de dades és excloent (podeu fer servir mètodes o blocs `synchronized`).

Només el servidor podrà accedir a la base de dades. Per tant, podríeu començar per moure el codi del client cap a una nova carpeta. En aquesta carpeta posaríeu la classe `Client` i una còpia de les classes `CharacterInfo` i `PackUtils`.

Quan l'usuari esculli una opció, l'heu d'enviar al servidor (podeu fer servir `writeInt` i `readInt`) juntament amb la informació necessària per a la seva execució. El servidor us enviarà una resposta (el format dependrà de cada cas i de l'èxit o no de la comanda), que imprimireu.

Per la seva banda, el servidor esperarà les peticions, les executarà i enviarà les respostes. Si voleu, podeu fer que imprimeixi informació per facilitar la depuració del codi.

1 - Llista tots els noms

El client envia l'opció 1. El servidor ha de respondre amb la llista dels noms. Primer enviarà el nombre de personatges i, tot seguit, tants `String`s com noms (feu servir `writeUTF` i `readUTF`).

2 - Obté la informació d'un personatge

Després de demanar el nom a l'usuari, el client enviarà l'opció 2 i el nom. El servidor cercarà el nom a la base de dades, enviarà un missatge indicant si l'ha trobat o no (si opteu per fer servir un booleà, podeu usar `writeBoolean` i `readBoolean`), i, en cas afirmatiu, enviarà la informació del personatge convenientment empaquetada (amb `toBytes`). El client rebrà el missatge que indica si el personatge està a la base de dades. Si hi és, rebrà la informació del personatge, la desempaquetarà i la mostrarà. En cas contrari mostrarà el missatge de "`Personatge no trobat.`".

3 - Afegeix un personatge

Després de demanar a l'usuari les dades del personatge i crear l'objecte `CharacterInfo` corresponent, el client enviarà l'opció 3 i la informació empaquetada. El servidor rebrà aquesta informació i intentarà inserir el personatge. Respondrà amb un missatge indicant si ho ha pogut fer. En cas negatiu, el client imprimirà "`Aquest personatge ja estava a la base de dades.`".

4 - Elimina un personatge

Després de demanar el nom a l'usuari, el client enviarà l'opció 4 i el nom. El servidor intentarà esborrar aquest personatge. Respondrà amb un missatge indicant si ho ha pogut fer. En cas negatiu, el client haurà d'imprimir "`Personatge no trobat.`".

5 - Sortir

Si el vostre servidor és concurrent i teniu la connexió oberta tota l'estona, el client enviarà l'opció 5, tancarà la connexió i sortirà del programa. El servidor, en rebre l'opció 5 també tancarà aquesta connexió, però no sortirà, ja que pot atendre altres clients.
Si heu fet un servidor interactiu, en aquest punt la connexió hauria d'estar tancada, per tant el client pot sortir directament sense notificar res al servidor.

El servidor no té per què incloure cap mecanisme de sortida. La idea és que es quedi executant indefinidament fins que no el mateu amb un ^C o un `kill`.

Avaluació:

- La pràctica s'ha d'implementar en llenguatge Java i s'ha d'executar correctament en una plataforma Linux.
- La pràctica es lliurarà via Campus Virtual (CV), dins l'apartat Activitats.
- Es recomana posar comentaris dins els fitxers .java que ajudin a interpretar el codi.
- La pràctica s'ha de resoldre individualment o en grups de màxim 2 persones.
- Com a comentaris de l'activitat heu d'indicar si la pràctica s'ha realitzat de forma individual o en grup. A més heu d'indicar els membres que composen el grup. Finalment, comenteu breument l'estratègia emprada en la implementació.

Aspectes a tenir en compte:

Alguns aspectes que heu de tenir en compte a l'hora de realitzar les vostres pràctiques i que **puntuaran negativament** si no els teniu en compte, **encara que la pràctica funcioni**:

- Nitidesa en el codi (tabulació correcta, no fer càlculs innecessaris, utilització correcta dels recursos de la màquina...).
- Utilitzar estructures algorísmiques adients per als problemes a resoldre.
- Utilitzeu noms de variables entenedors.
- Llegiu atentament l'enunciat i no implementeu funcionalitats diferents a les que us demana.
- En cas de realitzar pràctiques de forma "col·laborativa" entre diferents grups, esmentar-ho en el moment de l'entrega o en els comentaris, encara que finalment s'entreguin les pràctiques per separat o individualment.
- Si es detecta que la pràctica és copiada, la nota és un 0, tant pel que copia com pel que deixa copiar.

Per poder enviar tots els fitxers sense problemes els podeu comprimir utilitzant la comanda '`tar`' de la següent forma:

Suposant que teniu el codi dins la carpeta '`prac2`' podeu fer:

```
cd prac2
tar cvzf prac2.tar.gz *
```

El resultat és el fitxer `prac2.tar.gz` que conté tots els fitxers de la carpeta '`prac2`'.

Abans de comprimir la carpeta, esborreu els fitxers `.class`.