Съвременни Java технологии 2017/2018

Указания за разработка на курсов проект

версия 1.0

1. Проектите са индивидуални

2. Може да изберете един от 4-те предложени проекта или да измислите и предложите свой (приема се след наше одобрение - изпратете предложението си на stoyan.vellev@sap.com)

3. Всички проекти трябва да покриват следните общи условия:

   - Демонстрират знания за
     - вход-изход с файлове
     - паралелно изпълняващи се нишки
     - мрежова комуникация
   - Имат unit тестове с поне 30% line code coverage.
   - Имат валидация на входните данни
   - Имат добър обектно-ориентиран дизайн.
   - Използват подходящи изключения за докладване на грешки.
   - Използват подходящи структури от данни и ефективни алгоритми.
   - Имат четлив и добре структуриран код.

ЗАДАНИЯ ЗА КУРСОВ ПРОЕКТ

ТЕМА 1: Чат сървър

Да се разработи чат клиент-сървър със следната функционалност:

- Да обслужва много потребители едновременно.
- Всеки потребител има уникално име и парола, с което се автентицира пред системата. Данните за потребителите да се пазят във файл.
- Потребителите могат да напуснат системата по всяко време.
- Потребителите могат да изпращат лични съобщение до друг активен (регистриран и в момента свързан към сървъра) потребител.
- Комуникации чрез чат-стаи
    - Създаване и изтриване на чат-стая. Изтриването да е възможно само от потребителя, създал стаята.
    - Влизане и напускане на чат-стая
    - Показване на всички чат-стаи
    - Извеждане на всички активни потребители в дадена чат-стая
    - Изпращане на съобщение към всички в чат-стаята.
- Запазване на история на комуникацията в чат-стаи във файл.
    - При влизане в стая да се покаже историята на изпратените съобщения до момента (ако има такива)
- Потребителите да могат да трансферират файлове помежду си
    - Изпращащият посочва до кого иска да изпрати файл и къде се намира файлът
    - Получаващият получава нотификация, че някой се опитва да му изпрати файл и има възможност да приеме/откаже файла. Ако приеме файла, трябва да посочи къде да го съхрани.

Примерни клиентски команди:

- `connect <host> <port>` - свързва се към сървъра
- `register <username> <password>` - регистрира потребител
- `login <username> <password>` - влиза в системата
- `disconnect` – напуска системата
- `list-users` – извежда списък с всички активни в момента потребители

- `send <username> <message>` - изпраща лично съобщение до даден потребител.
- `send-file <username> <file_location>` - изпраща файл до потребител.

- `create-room <room_name>` - създава нова стая

- `delete-room <room_name>` - изтрива съществуваща стая
- `join-room <room_name>` - присъединява се към стая
- `leave-room <room_name>` - напуска стая
- `list-rooms` - извежда всички активни стаи (с поне 1 активен потребител)
- `list-users <room>` - извежда списък с всички активни потребители в дадена стая

TEMA 2: Game сървър за Battleships Online

Правила на играта

- Играта се играе от двама играчи.
- Всеки играч разполага с игрално поле, което се състои от 10x10 клетки. Редовете са обозначени с буквите от A до J, а колоните са номерирани с числата от 1 до 10.
- Всеки играч има на полето си:
    - 1 кораб, състоящ се от 5 клетки;
    - 2 кораба, състоящи се от 4 клетки;
    - 3 кораба, състоящи се от 3 клетки;
    - 4 кораба, състоящи се от 2 клетки;
- В началото на играта, всеки играч разполага корабите си на полето, като те могат да са само в права линия (хоризонтално или вертикално)
- Целта на всеки играч е да уцели корабите на противника си, като играчите се редуват и всеки има право на един изтрел на ход.
    - Играчът на ход подава координатите на клетката, по която стреля, и като отговор получава индикация дали е уцелил или не и ако е уцелил, дали корабът е потопен.
    - За да е потопен даден кораб, трябва да са уцелени всичките му клетки.
- Играта приключва, когато някой от играчите остане без кораби.

Game Server

Да се създаде Game Server със следните функционалности:

- Създаване на игра
- Извеждане на списък с всички игри, активни в момента, с информация дали играта е започнала и броя на играчите в нея.
- Присъединяване към вече създадена игра (всяка игра трябва да има уникален идентификатор), ако има свободно място.
- Присъединяване към случайна игра, в която има място.
- Запазване на състоянието на играта, в която сме в момента.
- Извеждане на всички запазени игри, в които сме участвали.
- Възстановяване на запазена игра и присъединяване към нея.
- Изтриване на запазена игра.

Game Client

Да се създаде клиент за сървъра, който има конзолен интерфейс

Примерен интерфейс

- Създаване на игра

```
$ java run-client.java --username gosho

// извеждане на възможните команди

Available commands:
       create-game <game-name>
       join-game [<game-name>] // ако липсва името, присъединяване към
случайна игра.
       saved-games
       load-game <game-name>
       delete-game
       ...

menu> create-game my-game
Created game "my-game", players 1/2
```

- Присъединяване към игра

```
$ java run-client.java --username tosho

// извеждане на възможните команди

menu> list-games
| NAME     | CREATOR | STATUS      | PLAYERS |
|----------+---------+-------------+---------|
| my-game  | pesho   | pending     | 1/2     |
| my-game-2| gosho   | in progress | 2/2     |

menu> join-game my-game
Joined game "my-game"
PLAYERS: 2/2, type "start" to start the game
```

- Въвеждане на ход

```
          YOUR BOARD
   1 2 3 4 5 6 7 8 9 10
  _ _ _ _ _ _ _ _ _ _
A |_|#|_|O|_|_|_|#|#|_|
B |_|#|_|_|_|_|_|_|_|_|
C |_|#|_|_|_|_|_|_|_|_|
D |_|X|_|_|#|#|#|_|O|_|
E |_|_|_|_|_|_|_|_|_|_|
F |_|_|O|_|_|_|O|_|_|_|
G |_|_|_|_|_|_|_|_|_|_|
H |_|#|_|_|_|X|X|X|X|_|
I |_|#|_|_|_|_|_|_|_|_|
J |_|#|_|_|_|_|_|_|_|_|
```

```
Legend:
            # - ship field
            X - hit ship field
            O - hit empty field
```

```
      ENEMY BOARD
   1 2 3 4 5 6 7 8 9 10
  _ _ _ _ _ _ _ _ _ _
A |_|_|_|_|_|_|_|_|_|_|
B |_|_|_|_|_|_|_|_|O|_|
C |_|O|_|_|_|_|O|_|_|_|
D |_|_|_|_|_|_|_|_|_|_|
E |_|_|_|O|_|_|_|_|O|_|
F |_|_|_|_|_|_|_|_|_|_|
G |_|_|_|_|_|_|_|X|_|_|
H |_|_|X|X|X|_|_|X|_|_|
I |_|_|_|_|_|_|_|X|_|_|
J |_|_|_|_|_|_|_|X|_|_|
```

```
gosho's last turn: D9
Enter your turn:
```

TEMA 3: Peer-to-peer file exchange (torrent сървър)

Да се имплементира опростен вариант на peer-to-peer система за обмен на файлове, която използва централен сървър за откриване на потребители и данни.

Системата трябва да се състои от две части:

1. Сървър, който съхранява метаданни за наличните файлове.
2. Клиенти, които теглят файлове от други клиенти – клиентите взимат информация от сървъра за това къде (от кой клиент) даден файл може да бъде свален .

Системата предоставя функционалност за сваляне на файлове от различни потребители. Всеки потребител може да сваля файл от всеки друг. Има централен сървър, който съхранява информация за потребителите и файловете, които могат да бъдат свалени от тях (файлове не се свалят и съхраняват на централния сървър).

Сървърът трябва да може да работи с много клиенти едновременно.
Сървърът съхранява информация за активните потребители в паметта (изберете подходяща структура от данни) – имената, адресите и портовете им (p2p обмен изисква да се знаят адресите на peer-ите); файловете, които могат да бъдат изтеглени от тях (абсолютен път). Тази информация се обновява, когато клиент регистрира файлове за изтегляне (команда **register** по-долу) и когато клиент затвори връзката със сървъра.

Клиентите могат да се свързват един с друг (peer-to-peer communication). За целта е нужно всеки да реализира „мини сървър“ при себе си. Този мини-сървър трябва да може да обработва командата `download`, описана по-долу. Можете да изберете дали мини-сървърът да обработва само една заявка за изтегляне или много паралелни заявки (ако е една - докато тя бива обработвана, оставащите заявки за изтегляне към мини-сървъра трябва да чакат).

Клиентът също така съхранява при себе си съответствие <username – user IP:port>.

Пример:

```
Pesho123 – 127.0.0.1:1234
Gosho321 – 127.0.0.1:2314
```

За да получи клиентът тази информация, той регулярно (през 30 секунди) пита сървъра за регистрираните потребители и техните адреси. Данните се записват във файл в указния по-горе формат.

Клиентът изпълнява следните команди:

1. `register <user> <file1, file2, file3, …., fileN>` - позволява на клиентите да „обявят“ кои файлове са налични за сваляне от тях. Чрез параметъра `username`, потребителят може да зададе свое уникално име (името, с което сървърът ще асоциира съответното IP).

2. `unregister <user> <file1, file2, file3, …., fileN>` - потребителят обявява, че от него вече не могат да се свалят файловете `<file1, file2, file3, …., fileN>`.

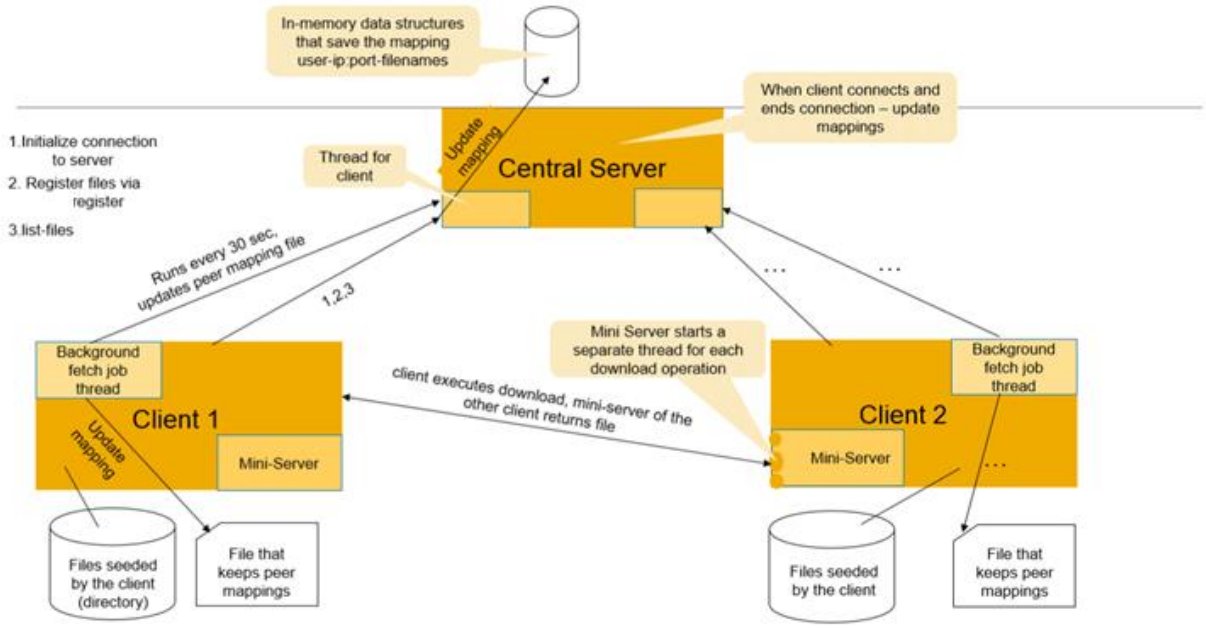
Забележка: За простота, не се интересуваме от security аспектите на решението, т.е. не е нужно да реализирате автентикация, която да гарантира, че даден потребител може да отрегистрира само файловете, които самият той е регистрирал.

3. `list-files` – връща файловете, налични за изтегляне, и потребителите, от които могат да бъдат изтеглени.
4. `download <user> <path to file on user> <path to save>` - изтегля дадения файл от съответния потребител.

Сървърът трябва да може да обработва изброените команди по подходящ начин:

1. При получаване на `register <user> <file1, file2, file3, …., fileN>` сървърът обновява информацията за този потребител (добавя информация, че тези файлове са налични за сваляне от съответния потребител).
2. При получаване на `unregister <user> <file1, file2, file3, …., fileN>` обновява информацията за този потребител (съответните файлове вече не са налични за сваляне от този потребител).
3. При получаване на `list-files`, връща регистрираните в сървъра файлове във формат `<user> : <path to file>`
4. При изпълняването на `download <user> <path to file on user> <path to save>`, не се случва комуникация с централния сървър.
    a. Клиентът определя IP адреса и порта на потребителя, от който може да бъде изтеглен даденият файл (от локалния си mapping).
    b. Клиентът изпраща командата на мини-сървъра на потребителя, определен в стъпка (a)
    c. Мини-сървърът изпраща файла
    d. След като потребителят получи файла, автоматично изпълнява командата `register <user> <path to saved file>`. По този начин информацията в главния сървър за потребителите, притежаващи този файл, се обновява.

Една подсказваща диаграма:



TEMA 4: Internet Movie Database search engine

IMDb (или Internet Movie Database) представлява онлайн база данни за филми, сериали, актьори и други. IMDb е най-големият сайт за въпросната тематика, като в него се съхраняват над 2 милиона статии за филми и телевизионни сериали. Съществува публичен програмен интерфейс (API), чрез който тези данни могат да бъдат достъпвани (безплатно до 1000 заявки на ден). Този интерфейс се нарича OMDb API и за целта на този курсов проект ще използваме него.

Да се имплементира клиент-сървър приложение, което предоставя функционалности за търсене в IMDb посредством OMDb API-то.

IMDb Server

- Да се имплементира многонишков сървър, който слуша на определен порт за връзки от клиентите.
- Сървърът получава команди от клиентите и връща подходящ резултат.
- Сървърът извлича необходимите му данни от OMDb API-то по HTTP и запазва (кешира) резултата в локалната файлова система. Например, при получаване на командата `get-movie Titanic`, сървърът прави HTTP GET заявка към http://www.omdbapi.com/?t=Titanic (OMDb API-то) и получава HTTP response със статус код 200 и с тяло следния JSON:

  {
  "Title": "Titanic",

```
    "Year": "1997",
    "Rated": "PG-13",
     …
}
```

Сървърът записва получения JSON във файл на локалната файлова система, като създава нов файл за всеки филм.

- При получаване на заявка, сървърът първо трябва да провери дали в кеша вече съществува файл с подаденото име за филм и ако е така, директно да върне съдържанието на файла, вместо да направи нова заявка към OMDb.

IMDb Client

- Клиентът осъществява връзка с IMDb Server на определен порт (напр. 4444), чете команди от стандартния вход и ги изпраща на сървъра.
- Да се имплементира команда `get-movie <movie_name> --fields=[field_1, field_2]`, която принтира подадените полета (`fields`) на филма с даденото заглавие (`<movie_name>`) в четим формат (например като JSON). Параметърът `fields` не е задължителен и може да липсва. При липсващ филм с подаденото заглавие, да се изведе подходящо съобщение за грешка.
- Да се имплементира команда `get-movies --order=[asc|desc] --genres=[genre_1, genre_2] --actors=[actor_1, actor_2]`. Параметрите `order` и `genre` не са задължителни и могат да липсват, или може да се подаде и само един от тях.
    - `order` – указва реда, в който филмите трябва да бъдат сортирани по полето imdbRating.
    - `genres` – филтрира филмите по подадените жанрове.
    - `actors` – филтрира филмите по подадените актьори.
- Да се имплементира команда `get-tv-series <name> --season=<value>`, която принтира епизодите на сериал за подадения сезон.
- Да се имплементира команда `get-movie-poster <name>`, която сваля изображението от полето `Poster` за подадения филм в локалната файлова система на сървъра, след което файлът с изображението се трансферира до клиента.