UNIVERSIDADE da MADEIRA

2022/2023
Programação Avançada – Projeto prático 1

1. Introdução

Com este projeto espera-se que os alunos demonstrem o conhecimento adquirido no primeiro terço da disciplina de Programação Avançada. Assim espera-se a entrega de um projeto em JAVA que resolva o problema descrito abaixo de uma forma paralela. De uma forma mais específica, espera-se a utilização de:

- Threads
- Semáforos
- Trancas
- Padrões arquiteturais para o desenvolvimento paralelo

2. Descrição do problema

O projeto proposto, definido de agora em diante como "**pa-chat-room**", tem como objetivo a criação de um servidor de mensagens entre vários clientes, mensagens essas que deverão ser transmitidas através de *broadcasting*. O projeto deverá ter como ponto de partida o código partilhado no repositório da disciplina[1].

De uma forma geral, o **pa-chat-room** deverá ser composto por um servidor, ao qual um ou mais clientes se poderão ligar. A implementação deverá permitir que, quando um cliente envia uma mensagem para o servidor, *todos* os clientes ligados ao servidor nesse instante receberão essa tal mensagem.

A maior prioridade deste programa (e, por consequência, do servidor) é garantir a integridade e a disponibilidade dos dados. Assim, deverão ser tomadas algumas decisões de implementação para que o servidor esteja sempre disponível. Então, o servidor deverá, por exemplo, criar *threads* filhas para todas as tarefas que potencialmente poderiam bloquear a *thread* principal.

O programa possui um ficheiro `server/server.log`, que deverá guardar todas as interações que aconteceram durante a execução: mensagens, entrada e saída de clientes, clientes à espera de vaga e *timestamps* associados.

Todas as configurações que considere necessárias para o servidor deverão ser lidas do ficheiro `server/server.config`, partilhado no repositório do projeto base. Toda a parametrização adicional que cada grupo ache necessária deverá ser adicionada a este ficheiro de configuração.

Na prática, serão avaliados os seguintes requisitos de implementação. Estes requisitos terão de ser cumpridos para a aprovação do projeto:

2.1 Requisitos de implementação

1. Terão de ser integrados **pelo menos 2 padrões** de desenvolvimento paralelo discutidos nas aulas para a resolução do problema acima.

[1] Note-se que terá de fazer ***download*** do repositório (ou seja, menu *Code* > *Download ZIP*). **Não faça** *clone* ou *fork* do mesmo.

2. O servidor deverá suportar um número máximo de clientes. Este valor deverá ser definido no ficheiro de configuração

    a. Caso o servidor esteja "cheio", os novos clientes deverão aguardar por uma vaga. Quando a vaga fica disponível (algum cliente desligou-se), os clientes em espera entram no servidor por ordem de chegada (FIFO).

3. Terão de ser utilizadas *Threads*/*Runnables*, Trancas e Semáforos.

4. É da responsabilidade do cliente escrever **todas** as suas interações no **pa-chat-room** no ficheiro `server.log` partilhado. Este ficheiro deverá registar as seguintes ações: (a) entradas e saídas no servidor; (b) todas as mensagens enviadas por qualquer cliente e (c) se o cliente se encontra em espera para entrar no servidor.

    a. O programa tem de garantir a integridade deste ficheiro. Apenas uma *thread* pode aceder (e escrever) ao mesmo de cada vez.

        i. Nestes casos os clientes deverão esperar a sua vez para escrever no ficheiro. Poderão existir situações em que o cliente já não está ligado ao servidor, mas continua à espera para finalizar a escrita no *log*.

    b. As mensagens de *log*, a serem guardadas no ficheiro `server.log`, deverão seguir o seguinte formato:

<timestamp> - Action : <tipo de ação> - <ID do cliente> - <mensagem (quando aplicável)>

```
2023-03-03T11:25:09+00:00 - Action : DISCONNECTED - CLIENT3
2023-03-03T11:26:09+00:00 - Action : MESSAGE - CLIENT10 - "olá mundo"
2023-03-03T11:27:09+00:00 - Action : WAITING - CLIENT20
2023-03-03T11:28:09+00:00 - Action : MESSAGE - CLIENT11 - "Cheguei"
2023-03-03T11:29:09+00:00 - Action : MESSAGE - CLIENT12 - "Olá"
2023-03-03T11:30:09+00:00 - Action : CONNECTED - CLIENT20
```

Figura 1: Exemplo de mensagens de *log*.

5. Antes das mensagens serem apresentadas no servidor, as mesmas deverão passar por um filtro. Este filtro, antes de apresentar as mensagens, remove das mesmas todas as ocorrências de determinadas palavras.

    a. A lista de palavras a remover deverá ser definida no ficheiro `filtro.txt` localizado em `server/filtro.txt`.

    b. O filtro deverá correr de forma completamente paralela ao envio e apresentação de mensagens no servidor.

6. A utilização do Git deverá observar as boas práticas apresentadas na TP1, nomeadamente, o desenvolvimento colaborativo (por exemplo, através da divisão dos requisitos pelos elementos do grupo), os *commits semânticos*, e a criação e a fusão de *branches* (onde cada *branch* deverá possuir um nome apropriado).

2.2 Requisito extra

Deverá ser possível ao utilizador alterar as configurações do servidor em tempo real (durante a execução), nomeadamente:

a. Alterar o número máximo de clientes suportados pelo servidor.
b. Adicionar e/ou remover palavras ao filtro de mensagens.

Qualquer uma destas alterações em tempo real **não deverá** resultar em qualquer perca de dados

3. Requisitos de entrega

1. O projeto deverá ser entregue até ao **dia 27 de Março de 2023**. A entrega deverá ser realizada através do repositório da disciplina, através da criação de uma *branch* com o nome "final".
2. Juntamente com a implementação, deverão ser entregues testes unitários e relatório de *code coverage* gerado pela biblioteca *jacoco* (já incluída no `pom.xml` do projeto partilhado).
3. Projetos que não compilem, e cuja sua execução seja, assim, impossível não serão considerados para avaliação.
4. Todas as classes, e os métodos do programa, deverão apresentar documentação de suporte de acordo com as práticas de documentação JAVA (Javadoc) discutidas na aula TP2

De forma a garantir os requisitos n.º 2, 3 e 4 é obrigatório que os alunos implementem GitHub Actions para a execução dos testes, criação do relatório de *code-coverage,* e criação da documentação de suporte. Poderão seguir os exemplos criados na aula TP2.

4. Avaliação Individual

A contribuição individual de cada aluno será avaliada tendo em conta o histórico de atividade do mesmo no repositório do projeto. Antes de iniciar a implementação, **todos** os grupos deverão criar os seus repositórios privados e partilhá-los com os docentes da disciplina (**Dntfreitas** e **Lipegno**).

5.Código de ética e honestidade académica

Nesta disciplina, espera-se que cada aluno subscreva os mais altos padrões de honestidade académica. Isto significa que cada ideia que não seja do aluno deve ser explicitamente creditada ao(s) respetivo(s) autor(es). O não cumprimento do disposto constitui uma prática de plágio. O plágio inclui a utilização de ideias, código ou conjuntos de soluções de outros alunos ou indivíduos, ou qualquer outra fonte para além dos textos de apoio à disciplina, sem dar o respetivo crédito a essas fontes. A menção das fontes não altera a classificação, mas os alunos não devem copiar código de outros colegas, ou dar o seu próprio código a outros colegas em qualquer circunstância. De notar que a responsabilidade de manter o acesso ao código somente para os colegas de grupo é de todos os elementos.