Enunciado de PCD - 2025/26
IsKahoot: jogo concorrente e distribuído

Versão 2 *- 29 de outubro de 2025

Resumo

Pretende-se desenvolver como projeto de PCD um jogo distribuído, aqui designado *IsKahoot*, uma versão adaptada do jogo popular Kahoot![1]. Apelando ao caráter lúdico juntamente com o educativo deste tipo de jogos, pretende-se assim motivar os alunos a dedicar o esforço necessário à aprendizagem dos conteúdos da cadeira.

Para bem servir os propósitos de aprendizagem, as características do jogo foram adaptadas, por forma a serem um bom cenário de aplicação de questões habituais no domínio da concorrência. Optou-se também por adicionar funcionalidades que modificam o comportamento normal do jogo, para lhe impor um caráter distribuído.

1 Descrição genérica do projeto

O jogo será disputado individualmente, com uma interface distinta e independente para cada jogador, em equipas de dois, sendo a contabilização de pontos e a classificação feita por equipa.

Cada ronda do jogo segue uma sequência estruturada de interações entre o servidor e os clientes, que representam os jogadores:

- Configuração do jogo: o servidor aceita comandos para criar um jogo, configurado pelo número de equipas, jogadores por equipa e perguntas, e gera um código único para identificar o jogo. Podem coexistir vários jogos simultaneamente, com funcionamento autónomo. Os clientes

*__Nota:__ o enunciado pode ser alterado para melhorar a clareza ou adequar-se melhor às necessidades de aprendizagem. Qualquer nova versão será publicada no moodle e será feita uma notificação por email.

[1] `https://en.wikipedia.org/wiki/Kahoot!`

entram no jogo inserindo os códigos do jogo e da equipa (a definir livremente pela própria equipa), bem como o identificador do jogador, e o servidor regista-os como participantes. Quando todos os jogadores se tiverem ligado ao jogo, o servidor seleciona o número escolhido de perguntas e inicia o jogo.

- Envio das Perguntas: O servidor envia a questão atual e as suas opções a todos os clientes e é criada uma contagem decrescente para terminar a ronda.
- Envio de Respostas : Os clientes enviam respostas logo que escolhidas pelo utilizador.
- Conclusão da Ronda: A ronda termina quando todos os jogadores ativos enviarem respostas ou o tempo expirar, conforme o que ocorrer primeiro. O servidor processa as respostas recolhidas, atualiza as pontuações e gera as estatísticas da ronda, que envia a todos os clientes, que as devem exibir na GUI.
- Próxima Pergunta: O servidor avança para a questão seguinte, repetindo o fluxo até que todas as questões tenham sido respondidas.
- Fim do Jogo: Após a ronda final, o servidor envia as pontuações finais e termina as *threads* locais do jogo.

1.1 Responsabilidades do Servidor

O servidor armazena todas as perguntas bem como as suas respostas, gere a lógica do jogo e monitoriza os jogos ativos e os jogadores neles participantes. Cada jogador ligado ao servidor está associado a uma *thread* `DealWithClient` que gere as interações com ele, incluindo o envio e a receção de respostas.

1.2 Responsabilidades do Cliente

No arranque do cliente, este liga-se automaticamente ao servidor, indicando a identificação do jogo e equipa a que pertence, bem como um identificador individual. Estes parâmetros, bem como o endereço e porto do servidor, devem ser recebidos como argumentos de execução do *main*[2].

As principais interações executadas pelo cliente serão feitas através da GUI: exibição de perguntas, cronómetros e placares recebidos, bem como enviar respostas durante cada ronda.

[2]https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html

1.3 Estatísticas do jogo

Cada jogo é representado por um objeto *GameState* no servidor. Este objeto define a pergunta atual, implementa o cronómetro decrescente da ronda, e mantém registo dos jogadores e equipas para as respostas enviadas e o placar.

Para evitar bloqueios desnecessários, é comum separar as responsabilidades dentro do *GameState* em vários componentes, permitindo que diferentes aspetos do jogo (por exemplo, atualizações do placar, recolha de respostas, gestão do cronómetro) sejam acedidos e modificados de forma independente.

1.4 Placar de desempenho

Após cada ronda, o servidor envia a todos os jogadores no jogo um placar atualizado mostrando a posição atual da equipa, os pontos ganhos na ronda, a pontuação acumulada e a lista de todas as equipas (e respetivas pontuações) do jogo.

2 Requisitos do projeto

O projeto é uma ferramenta de aprendizagem, pelo que deve servir para adquirir conhecimentos em vários tópicos do programa de PCD, que passamos a elencar e que devem ser escrupulosamente seguidos.

Todos os mecanismos de coordenação devem ser desenvolvidos pelo próprio grupo, não devendo ser usados os equivalentes disponibilizados nas bibliotecas padrão do Java.

2.1 Nome de utilizador, identificação de equipa e jogo

Ao iniciar a aplicação, o utilizador recebe como parâmetro de execução um identificador de utilizador, que deve ser único em todo o sistema. Se outro jogador já estiver a utilizar o nome escolhido, o servidor rejeita-o liminarmente. Este nome de utilizador, bem como a identificação do jogo e equipa devem ser enviados ao servidor aquando da ligação inicial. Se o jogo enviado não existir, ou se estiver a exceder o número de equipas previsto, a ligação é recusada. O endereço e porto do servidor, bem como os identificadores de jogo, equipa e jogador são dados como argumentos ao programa do cliente.
java clienteKahoot { IP PORT Jogo Equipa Username}

2.2 Códigos do jogo

Servidor deve ser capaz de gerar um código único que não esteja em utilização. O servidor deve conseguir armazenar os códigos e descartá-los quando não estão mais em utilização. A criação de um novo jogo é feita através de uma interface textual (TUI), que permitirá a qualquer momento a criação de um jogo e, opcionalmente, a visualização dos jogos existentes, incluindo as classificações e estado do jogo. Sugere-se o seguinte comando: `new` <num$_e quipas >< num_j ogadores_p or_e quipa >< num_p erguntas >

2.3 Tabela de pontuações

Após cada ronda, a pontuação das equipas é atualizada com base nos critérios indicados à frente, se a opção escolhida for a correta. No fim de cada pergunta, será apresentado um placar sumário que deve ser enviados para todos os jogadores.

2.4 Ciclo de perguntas

Após o servidor enviar a pergunta e as opções de resposta aos jogadores, este inicia uma contagem decrescente de, por exemplo 30 segundos. As perguntas devem ser lidas de um ficheiro de texto em formato json, conforme exemplo incluído mais abaixo. Ficheiros json são fáceis de processar com auxílio da biblioteca Gson[3], biblioteca de Java da Google que permite converter objetos em JSON e vice-versa.

2.5 Receção de respostas

Cada jogo armazena as respostas enviadas pelos jogadores à questão atual. Como vários jogadores podem enviar respostas em simultâneo, o servidor deve garantir que todas as respostas enviadas são registadas exatamente uma única vez, sem interferências. Quando todos os jogadores enviarem a sua resposta, ou se o tempo-limite se esgotar, a ronda termina. Assim que a ronda acaba é enviado um placar resumido aos clientes a mostrar o desempenho das equipas na ronda.

2.6 Coordenação para o fim do jogo

Como descrito na secção anterior, o jogo termina quando acabarem as submissões ou o tempo expirar na última pergunta. Quando tal

[3] https://mvnrepository.com/artifact/com.google.code.gson/gson

acontecer,o jogo termina para todos os participantes. Para implementar esta coordenação, devem ser interrompidos todas as *threads* dos jogadores em execução (DealWithClient).

2.7 Tipos de perguntas

Vão existir dois tipos de perguntas: individuais, em que cada jogador envia independentemente a sua resposta, e rondas de equipa, em que apenas se avaliam as respostas quando todas as respostas de uma equipa sejam recebidas (ou quando o tempo limite expirar). Ao longo do jogo as perguntas vão ser alternadamente individuais e de equipa.

Perguntas individuais Neste tipo de perguntas a classificação é calculada para cada jogador, somando-se os valores de todos os membros da mesma equipa.

Existirão pontuações bonificadas para os primeiros dois jogadores a responder: a sua pontuação será o dobro da cotação da pergunta. Sugere-se a utilização de uma CountDownLatch + Timer. Esse fator deve ser computado quando a resposta for submetida ao semáforo, em particular como valor de devolução do método countDown. A API desta classe será esta:

```
1
2 public class ModifiedCountdownLatch {
3         public ModifiedCountdownLatch(int bonusFactor, int
              ↪ bonusCount, int waitPeriod, int count);
4         public synchronized int countdown() ;
5         public synchronized void await() throws
              ↪ InterruptedException ;
6 }
```

Perguntas de equipa Nestas perguntas, a cotação das perguntas de uma equipa apenas serão decididas após a receção das respostas de todos os seus elementos (ou quando o tempo limite expirar). Caso todos acertem, têm a cotação da pergunta duplicada. Caso algum falhe, apenas será considerada a melhor pontuação de entre eles, sem nenhuma bonificação. Sugere-se a utilização de uma barreira para estabelecer esta coordenação. A implementação desta estrutura de coordenação deve fazer uso de variáveis condicionais.

2.8 *Threadpool:* tarefa extra

Para limitar a concorrência em cenários em que haja muitos jogos a decorrer, sugere-se a utilização de uma *Threadpool*, de maneira a limitar os jogos a decorrer a cinco.

3 Detalhes de implementação

3.1 Lançamento do servidor

O servidor é lançado sem argumentos, passando a estar preparado para criar novos jogos. Apenas quando for criado um jogo poderá o servidor receber ligações de clientes remotos.

3.2 Criação de novo jogo

A criação de um novo jogo é feita na sequência de um comando inserido na TUI do servidor. A partir desse momento, será possível receber ligações de jogadores e, quando estiverem ligados todos os jogadores previstos, iniciar o ciclo do jogo.

3.3 Ficheiro com as perguntas

É fornecido um ficheiro exemplo com perguntas sobre a matéria de PCD, que pode ser usado para alimentar o jogo. As perguntas serão usadas nas rondas de forma aleatória, e os grupos podem criar novas perguntas para aumentar o número de perguntas existente.

O formato do ficheiro será algo como:

```
{
  "quizzes": [
    {
      "name": "PCD-1",
      "questions": [
        {
          "question": "O␣que␣  ␣uma␣thread?",
          "points": 5,
          "correct": 3,
          "options": [
            "Processo",
            "Aplica  o",
            "Programa",
```

```
14              "Processo␣Ligeiro"
15            ]
16          },
17          {
18            "question": "Qual␣destas␣op  es␣n o␣  ␣um␣
                ↪ m todo␣bloqueante?",
19            "points": 3,
20            "correct": 2,
21            "options": [
22              "join()",
23              "sleep(<millis>)",
24              "interrupted()",
25              "wait()"
26            ]
27          }]
28      }
29  }
```

Uma vez que apenas existirá um *quiz*, o objeto quizzes (uma lista de *quiz* pode ser desprezado e eliminado do ficheiro.

3.4 Perguntas individuais: CountDownLatch modificado

Para poder classificar corretamente as perguntas individuais, deve ser aplicado um *CountDownLatch* alterado. Este deve ter um temporizador para detetar que passou o tempo limite, no método await, bloqueante, que apenas desbloqueará quando o *CountDownLatch* desbloquear. O método countdown deve ter um valor de devolução inteiro, que será o fator a aplicar à cotação da resposta, e que será invocado quando esta for recebida de um dos jogadores.

3.5 Perguntas por equipa: barreira

As perguntas por equipa devem ser coordenadas por uma barreira modificada, de maneira a apenas classificar a resposta da equipa quando todos os jogadores tiverem respondido, ou quando o tempo limite expirar. Se este tempo de facto expirar, todos as chamadas aos métodos await devem ser desbloqueadas, e o cálculo das pontuações deve ser feito através da funcionalidade barrierAction. Esta barreira deve ser implementada usando variáveis condicionais.

4 Fases de desenvolvimento

Nesta secção sugere-se um encadeamento de fases que permitem um desenvolvimento sustentado deste projeto.

1. Desenvolvimento da GUI: desenvolver a aplicação cliente que apresentará as perguntas aos utilizadores e receberá as respostas;
2. Desenvolvimento da estrutura GameState: desenvolvimento de uma estrutura adequada para manter todos os dados relevantes do estado do(s) jogo(s);
3. Leitura das perguntas a partir do ficheiro: processamento do ficheiro em formato json e carregamento para objetos *Java*.
4. Criação do servidor e ligação inicial dos clientes: desenvolver uma funcionalidade inicial de ligação no modelo cliente servidor;
5. Troca de mensagens entre clientes e servidor: desenvolvimento das classes necessárias ao envio de mensagens, através de canais de objetos, e sua aplicação na comunicação entre o servidor e clientes;
6. Processamento das respostas: aplicação do *CountDownLatch* e barreira desenvolvidos e contabilização dos pontos de cada equipa.
7. Desenvolvimento do ciclo e fim do jogo: criação do ciclo de jogo que encadeie as perguntas e respostas até ao final.
8. Aplicação da *Threadpool* : tarefa extra Esta fase é opcional.

5 Entregas

Para favorecer o início atempado do desenvolvimento do projeto, existe uma entrega intermédia. Pede-se que seja feita uma apresentação das 3 fases iniciais, referentes a aspetos ainda sem concorrência, no turno de laboratório da semana de 11 a 14 de novembro, onde deverá ser feita uma breve apresentação do trabalho feito. Esta entrega é obrigatória e eliminatória, e feita presencialmente.

A entrega final será às 09:00 de 15.12.2025, em página no moodle a disponibilizar oportunamente. As discussões finais, de que alguns alunos poderão eventualmente ser dispensados, decorrerão nos laboratórios

de 16 a 19 de dezembro e, se necessário, estender-se-ão eventualmente para os dias seguintes.

O modelo da avaliação é habitual nas cadeiras de programação: o projeto é obrigatório mas não conta diretamente para a nota, podendo apenas limitar a nota final, conforme critérios apresentados na primeira aula e constante da FUC.

6 Avaliação

Todas as fases comuns descritas na secção 4 devem ser implementadas, pois o projeto é primordialmente uma ferramenta de aprendizagem e essas fases correspondem a partes importantes da matéria a aprender e a ser avaliada.

As tarefas opcionais podem levar a uma avaliação maximizada do projeto.

Caso haja lacunas menores nesta implementação, os alunos poderão mesmo assim ter aprovação com uma valoração qualitativa mais baixa.

7 Histórico de versões

1 Versão original

2 Alteração da estrutura de coordenação das perguntas individuais de semáforo para *CountDownLatch*. Pedido de uso de variáveis condicionais na barreira. Sugestão de aplicação de *ThreadPool*.