universidade de aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Problema obrigatório 1

GAME OF THE ROPE

Jogo da corda

Curso	[8240] MI em Engenharia de Computadores e Telemática
Disciplina	[40814] Sistemas Distribuídos
Ano letivo	2015/2016

Alunos	[68021] Gabriel Vieira
	[68779] Rui Oliveira
Prática	P1
Docente	Professor Óscar Pereira

Aveiro, 5 de Abril de 2016

Conteúdo

1 Organização

De seguida iremos apresentar a organização em directórios do nosso programa.

- **main**: é declarada a main principal. É neste local que são instanciados todos os threads e monitores utilizados. Os threads são ativados através do método start().
- **domain**: existem 3 classes que representam cada entidade: Coach.java Contestant.java e Referee.java
- **states**: neste package existe três enumerados onde estão descritos os estados das três entidades presentes.
- **monitors**: existem três monitores, que irão controlar todos os processos envolventes de cada entidade. São eles: MBench.java, MPlayground.java, MSite.java. Para além dos monitores descritos anteriormente, existe o módulo MRepository.java que permitirá criar um ficheiro de log com todas as alterações ocorridas no processo de execução.
- **interfaces:** neste package estão presentes 11 interfaces(java) que permitirão uma maior modelação na instanciação de objetos. É nestas interfaces que são declarados os métodos utilizados por cada entidade, sendo estes implementados em cada monitor.

2 Repositório

https://github.com/ruipoliveira/gameoftherope.git

3 Tipo de dados

3.1 Entidade ativa: instanciado e com um fio de execução

3.1.1 Árbitro (Referee)

- **Estrutura de dados interna**

  state: variável onde é guardado o estado actual do arbitro.

- **Operações**

  setState(ERefereeState state) - permite atualizar o estado do arbitro

  RefereeState getCurrentState() - permite aceder ao estado do arbitro

3.1.2 Treinador (Coach)

- **Estrutura de dados interna**

  state: variável onde é guardado o estado actual do treinador.

- **Operações**

  setState(ECoachesState state) - permite atualizar o estado do treinador

  ECoachesState getCurrentState() - permite aceder ao estado do treinador

  int getIdCoach() - Permite aceder ao identificador do treinador.

3.1.3 Jogador (Contestant)

- **Estrutura de dados interna**

  state: variável onde é guardado o estado actual do jogador.

  contestStrength: variável onde é guardado a força de cada jogador. Esta força é obtida através da função generateStrength()

- **Operações**

  setState(EContestantsState state) - permite atualizar o estado do jogador

  EContestantsState getCurrentState() - permite aceder ao estado do jogador

  int getIdCoach() - Permite aceder ao identificador do treinador.

  int getStrength() - Permite aceder à força do respetivo jogador.

  int generateStrength() - Permite gerar de forma aleatória a força associada a cada jogador. Valor da força é um inteiro entre 10 e 20.

3.2 Entidade passiva: instanciado e sem um fio de execução

3.2.1 Banco (Bench)

- **Operações**

  callContestants(int coachId) - permite que os treinadores chamem os jogadores para campo. Nesta mesma função o treinador irá esperar que os jogadores estejam pronto no banco, de seguida irá escolher três deles para ir jogar (o método de escolha é aleatório). Seguidamente, o treinador irá esperar que os jogadores estejam prontos na corda.

callTrial(int numGame, int numTrial) - para o primeiro trial o arbitro irá informar os treinadores para se prepararem para um novo trial. No caso dos restantes trials, o jogador irá esperar que os treinadores terminem a operação reviewNotes().

isPlayerSelected(int coachId, int contestId) - verifica se um jogador foi escolhido para se deslocar para a corda.

reviewNotes(int coachId) - O treinador aguarda que os jogadores se sentem, caso isso aconteça a estrutura de dados que armazena os jogadores na corda é limpa.

seatDown(int coachId, int contestId) - os jogadores esperam que os restantes jogadores já tenham acabado a simulação de puxar a corda, caso isso aconteça os jogadores irão sentar-se.

allSittingTeams() - verifica se todos os jogadores estão sentados.

followCoachAdvice(int coachId, int contestId) - os jogadores aguardam que sejam chamados pelo seu treinador para irem jogar um novo trial. No fim do mesmo acontecer, os jogadores irão informar o respetivo treinador que estão prontos.

endOfTheGame() - permite verificar quando é que um jogo acaba ou não.

3.2.2 Local do árbitro (Site)

- **Operações**

  announceNewGame(int numGame, int nrTrial) - funcao de transição de estado. Apenas é usada para guarda o numero do jogo e do trial neste monitor.

  declareGameWinner(int posPull) - permite definir qual das equipas é a vencedora do respetivo jogo. O resultado é obtido através da observação da posição da corda.

  declareMatchWinner - define qual das equipas é a vencedora do match. Verifica qual das equipas ganhou o maior numero de jogos.

  endOperCoach(int id) - permite verificar quando é que um dado treinador termina o seu processo.

3.2.3 Campo (Playground)

- **Operações**

startTrial(int nrGame,int nrTrial) - permite dar inicio a um novo trial. Inicialmente o arbitro irá esperar que os treinador estejam pronto e posteriormente irá informar os jogadores que podem iniciar a partida.

assertTrialDecision() - está função é executa pelo arbitro no final de cada trial. Inicialmente o arbitro aguarda que todos os jogadores tenham puxado a corda. Posteriormente, irá somar a força de todos os jogadores de cada equipa e acumulá-lo numa variável. Seguidamente irá decidir quem vence o trial, se fica empatado ou se existe knockOut.

informReferee(int coachId) - treinador informa arbitro que estão prontos.

getPositionPull() - retorna posição da corda

setPositionPull(int posPull) - atualiza posição da corda

getReady(int coachId, int contId) - jogadores esperam que o arbitro inicie o trial .

amDone(int coachId, int contId, int contestStrength) - nesta função é simulado o puxo da corda.

3.2.4 Repositório geral

- **Operações**

  initWriting() - escreve cabeçalho num ficheiro.

  writeLine() - escreve cada linha num ficheiro

  endWriting() - termina processo de inscrita no ficheiro.

  writeLineGame() - escreve primeira linha no inicio de cada jogo e repete cabeçalho.

  Funções de atualização de estados: -*updateRefState(ERefereeState state)* -*updateCoachState(int idCoach, ECoachesState state)* -*updateRefState(ERefereeState state)*

  updateStrength(int idTeam, int idContest, int contestStrength) - função de atualização da força de um determinado jogador. É usada no construtor de um jogador.

  updateStrengthAndWrite(int idTeam,int contestId, int contestStrength) - similar à anterior mas permite escrever uma nova linha num ficheiro.

  updatePullPosition(int posPull) - atualiza a posição atual da corda

  updateTrialNumber(int nrTrial) - atualiza o número atual do trial

  updateGameNumber(int nrGame) - atualiza o número atual do jogo

isKnockOut(int nrGame, int nrTrial, String team) - se existir knockOut esta função é invocada. Permitirá escrever uma linha indicando o mesmo.

isEnd(int nrGame, String team) - se o jogo terminar esta função é invocada. Permitirá escrever uma linha indicando o mesmo.

wasADraw(int nrGame) - se o jogo ficar empatado esta função é invocada. Permitirá escrever uma linha indicando o mesmo.

endMatch(String team, int resultA, int resultB) - se o match terminar esta função é invocada. Permitirá escrever uma linha indicando o mesmo.

4 Diagrama de interação

Foi considerado o esquema introduzido durante as aulas teóricas para a representação do diagrama de interação.

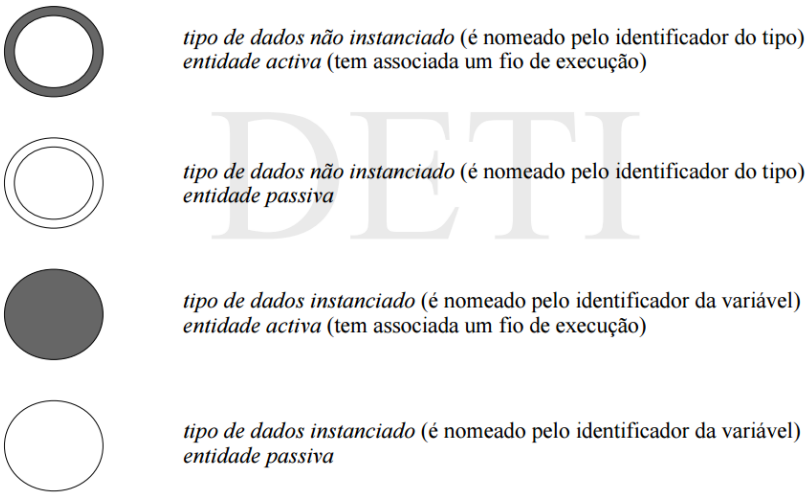

Figura 1: Elementos base dos diagramas de interação

Sendo assim o resultado é o seguinte:

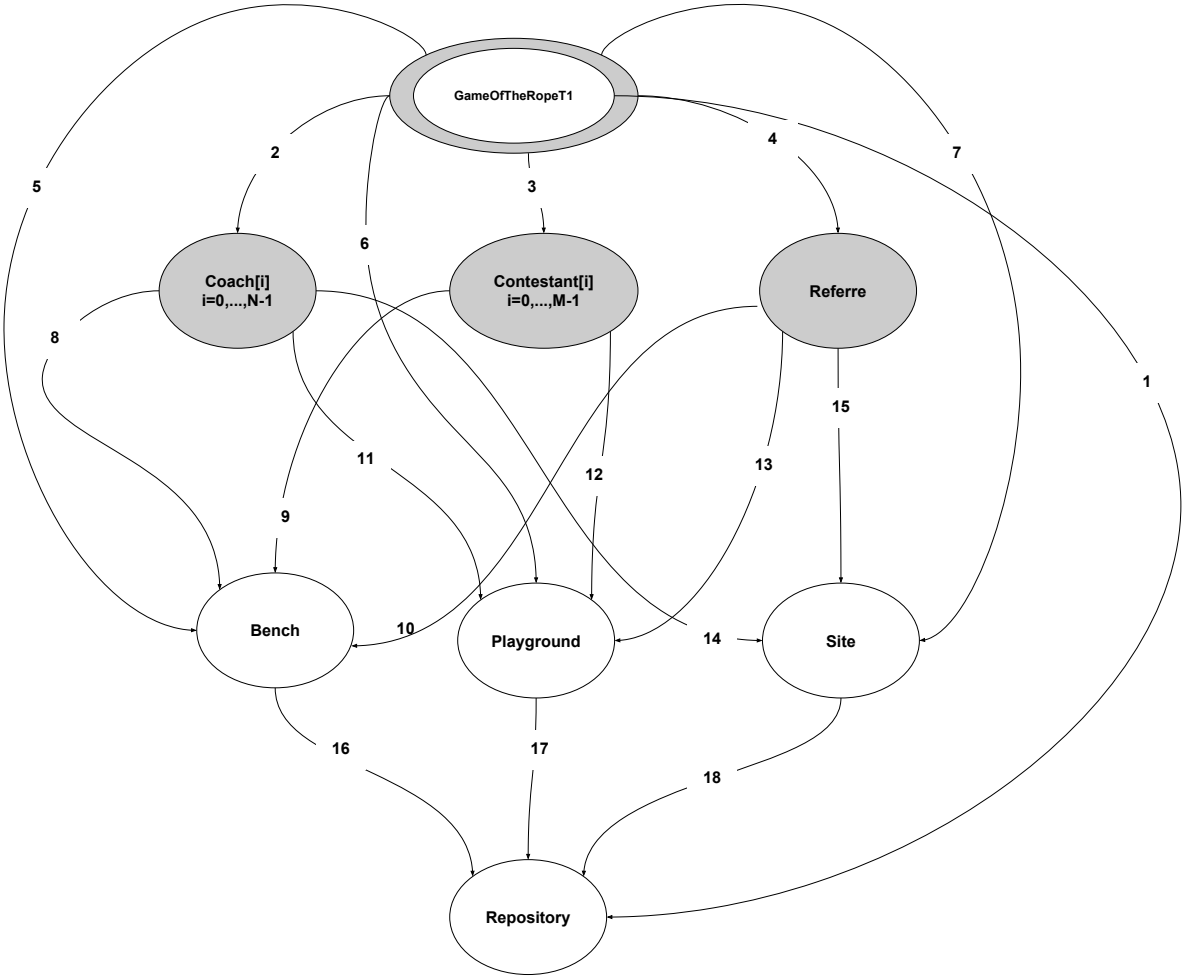

Figura 2: Diagrama de interação

1. instantiate
2. instantiate, start, join
3. instantiate, start, join
4. instantiate, start, join
5. instantiate
6. instantiate

7. instantiate
8. callContestants, reviewNotes, endOfTheGame
9. seatDown, followCoachAdvice, isPlayerSelected
10. callTrial, allSittingTeams
11. informReferee
12. getReady, amDone
13. startTrial, assertTrialDecision, getPositionPull, setPullPosition
14. endOperCoach
15. announceNewGame, declareGameWinner, declareMatchWinner
16. updateCoachState
17. updateContestState, updateStrengthAndWrite
18. updateGameNumber, updateTrialNumber, updateRefState, isKnockOut