Game Project

Estruturas de Informação

Pretende-se que desenvolvam uma biblioteca de classes, respetivos métodos e testes, que permitam gerir o funcionamento de um jogo de computador de estratégia. A biblioteca pretende-se suficientemente genérica para adaptar a vários jogos com características semelhantes, pelo que apenas implementa algumas funcionalidades de suporte, mas não a tomada de decisões.

A biblioteca gere diversos tipos de entidades, de acordo com um conjunto de regras, dos quais se apresentam as necessárias para implementar as funcionalidades solicitadas:

- Um **local** tem um nome e pode ter no máximo uma personagem associada, o seu "dono", mas podem existir locais sem personagens e uma personagem ser "dona" de vários locais. Um local tem também associado um valor de pontos (número inteiro), que representa a dificuldade de entrar no local.
- Os locais estão ligados entre si por **estradas**, sendo que cada estrada tem um valor de pontos (valor inteiro) associado à dificuldade de a percorrer, existindo assim uma rede de estradas.
- Uma **personagem** tem um nome e um valor de pontos (número inteiro) que representa a sua força. Duas personagens podem efetuar uma parceria (**aliança**), sendo que as alianças não são exclusivas (a personagem **A** pode estar aliada a **B** e **C**, podendo **B** estar, ou não, aliada a **C** e/ou a outras personagens), pelo que existe uma rede de alianças.
- As alianças podem ser públicas ou privadas (neste caso apenas os dois participantes na aliança a conhecem) e têm associado um fator de compatibilidade (número Float, entre zero e um). A força de uma aliança é igual à soma das forças dos 2 aliados multiplicada pelo fator de compatibilidade.
- Uma personagem para conquistar um determinado local **X**, tem que ser dona de um local vizinho (diretamente ligado por uma estrada), e ter força suficiente para percorrer a estrada, aceder ao local e, caso exista, derrotar o dono atual de **X (o valor de pontos da personagem tem que ser maior que a soma destes três valores)**. Uma personagem para conquistar um determinado local **X** não diretamente acessível tem que conquistar todos os locais entre **X** e um local do qual seja dono.
- Uma personagem pode decidir propor a uma das suas alianças conquistar/defender um determinado local **X**. Neste caso, a sua força será a força da sua aliança, e a redução de forças será aplicada metade a cada membro da aliança.
- Uma personagem **A** pode realizar uma nova aliança com outra personagem **B**. Se **B** não fizer parte da rede de alianças de **A**, então o fator de compatibilidade da aliança é gerado aleatoriamente. Se **B** fizer parte da rede de alianças de **A**, então o fator de compatibilidade é a média dos fatores de compatibilidade do menor caminho (em número de alianças) de **A** a **B**. De notar que para esse cálculo, **A** apenas pode considerar as alianças públicas (exceto as suas que pode usar as privadas).

Game Project

Usando as classes de manipulação de grafos apresentadas nas aulas, pretende-se que construam uma biblioteca simplificada que permita gerir a rede de locais e estradas bem como a rede de personagens e alianças. Todas as funcionalidades implementadas deverão usar um algoritmo eficiente e escalável.

1. O grafo dos locais e estradas deve ser implementado usando a representação **Matriz de Adjacências**, com as seguintes funcionalidades:

a) Construir o grafo a partir de um ficheiro de texto com a informação estruturada do seguinte modo (não existem espaços):

```
LOCAIS
NomeLocal1,Pontos
NomeLocal2,Pontos
…
NomeLocalN, Pontos

CAMINHOS
NomeLocal1,NomeLocal2,Dificuldade
…
NomeLocal2,NomeLocalN,Dificuldade
```

b) Apresentar o caminho com menor dificuldade (considerando apenas a dificuldade das estradas) entre dois locais.

c) Sem considerar alianças, verificar se uma determinada personagem pode conquistar um determinado local, devolvendo também a força necessária e a lista mínima de locais intermédios a conquistar, caso seja necessário.

2. As relações de aliança entre as várias personagens devem ser implementadas através de um grafo usando a representação **Map de Adjacências** com as seguintes funcionalidades:

a) Construir o grafo a partir de um ficheiro de texto com a informação estruturada do seguinte modo (não existem espaços), sendo que os fatores de compatibilidade deverão ser gerados aleatoriamente (True/False representa se a aliança é pública) e inicialmente atribua aleatoriamente cada personagem a um local:

```
PERSONAGENS:
NomePers1,Pontos
NomePers2,Pontos
…
NomePers2,Pontos

ALIANÇAS
NomePers1,NomePers2,TRUE
…
NomePers2,NomePersN,FALSE
```

b) Devolver uma lista com todos os aliados de uma dada personagem.

c) Determinar qual a aliança mais forte, retornando a força e as personagens dessa aliança.

d) Realizar uma nova aliança entre uma personagem A e uma personagem B.

e) Criar um novo grafo representando todas as alianças que podem ser realizadas entre todas as personagens, caso todas as alianças existentes fossem públicas.

3. Existe também uma classe controlo do jogo, o qual gere a relação entre as redes de locais e de alianças, com as seguintes funcionalidades (pode, se achar necessário acrescentar funcionalidades nas classes anteriores):

f) Verificar se uma personagem pode conquistar, junto com um dos seus aliados, um determinado local X (assuma que o dono de X, caso exista, não usa as suas alianças), devolvendo qual o aliado, assim como o valor necessário e a lista mínima de locais intermédios a conquistar, caso seja necessário. De notar que o aliado não pode ser dono de X nem de nenhum dos locais intermédios.

Normas

- A avaliação do trabalho será feita principalmente em função das classes propostas, nomeadamente em termos da sua conformidade com o Paradigma da Programação por Objetos e **eficiência** das estruturas de dados usadas e funcionalidades solicitadas.
- O trabalho deverá ser realizado em **grupos de dois alunos**. Os grupos têm de ser formados e enviados por *email* ao docente das aulas PL, até ao final da **1ª semana aulas**.
- O projeto tem de ser desenvolvido em Java e todas as funcionalidades testadas através de testes unitários e usando os ficheiros de teste disponibilizados.
- É obrigatório o uso da ferramenta de **controle de versões Git**.
- O relatório deverá ser elaborado para cada uma das partes do trabalho e deve servir de ferramenta de avaliação posterior à apresentação. Nele devem apresentar o digrama de classes, algoritmos dos métodos, análise de complexidade de todas as funcionalidades implementadas, melhoramentos possíveis.
- Cada Parte do trabalho deve ser submetida no Moodle até às **24 horas do dia 19 de Novembro**. A partir desta data a nota do trabalho será penalizada **10% por cada dia de atraso** e não se aceitam trabalhos **após dois dias** das datas indicadas.
- Na semana seguinte à data de entrega o professor das aulas práticas fará a cada grupo de trabalho uma avaliação qualitativa do projeto submetido.
- A apresentação/avaliação do trabalho final será individual e realiza-se na 13ª semana de aulas, em datas a fixar com o professor das aulas práticas.