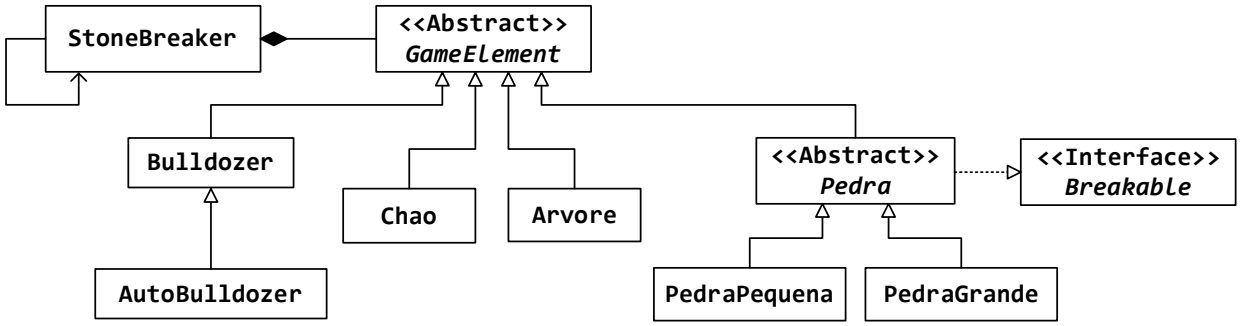
Revisões para o 2º teste intercalar de POO – 2021/22

Para resolver estes exercícios, deverá começar por importar para o eclipse o projeto `Exerc-revisoes2`, que contém código base. O projeto está disponibilizado na plataforma de e-learning junto a esta ficha.

Pretende-se desenvolver um jogo muito simples – o *StoneBreaker* – com mecânicas idênticas à do projeto *FireFight*. O objetivo do jogo resume-se a destruir as pedras que estão na área de jogo.

Descrição do jogo

No seguinte esquema apresenta-se um diagrama UML das classes (e interface) que compõem este jogo. A classe `StoneBreaker` é a classe que contém o motor de jogo e não é suposto ser alterada, exceto para realizar a última questão desta ficha.



O jogo funciona da seguinte maneira:

- O jogador controla um bulldozer, mas existem também bulldozers que se movimentam automaticamente (autobulldozers);
- Em cada jogada, todos os bulldozers se tentam movimentar;
- Os bulldozers não podem ir para cima das árvores (i.e., as árvores não são transponíveis);
- As pedras pequenas são destruídas (i.e., removidas do jogo) quando um bulldozer passa por cima delas;
- As pedras grandes não são transponíveis, mas "partem-se" quando um bulldozer tenta avançar para cima delas – nessa situação caso transformam-se em pedras pequenas e na jogada seguinte já podem ser destruídas;
- Os autobulldozers têm exatamente o mesmo comportamento do bulldozer controlado pelo jogador, mas movem-se aleatoriamente;
- O bulldozer e os autobulldozers não são transponíveis (i.e., não podem ir para cima uns dos outros).

Trabalho a desenvolver

1) Comece por analisar o código das classes `StoneBreaker` e `GameElement`, verificando os métodos que estão disponíveis, em particular os que podem ser invocados a partir das restantes classes. A classe `StoneBreaker` só é suposto ser alterada na questão 7; a classe `GameElement` pode ser alterada se assim o entender (embora à partida isso não deva ser necessário).

2) Complete a classe `Arvore`, sobrepondo o método `isTransposable()`, herdado da classe `GameElement`, de forma a que as árvores deixem de ser transponíveis.

3) Complete a classe `Bulldozer`, definindo os atributos e métodos necessários para realizar o movimento, interagir com os objetos `Breakable` que possam existir na posição de destino, mudar a imagem consoante a direção do movimento e definir a transponibilidade deste tipo de objetos.

4) Crie a classe `Pedra` e depois complete suas classes derivadas: `PedraPequena` e `PedraGrande`. Siga o diagrama UML.

5) Complete a classe `AutoBulldozer`, procurando minimizar a quantidade de código através do uso adequado de herança. Note também que é suposto os autobulldozers usarem uma imagem diferente do bulldozer.

6) Acrescente um método `void givePointsTo(Bulldozer b)` no interface `Breakable`. Depois disso acrescente o que for necessário para dar pontos ao bulldozer ou aos autobulldozers de cada vez que estes "partem" um objeto `Breakable`. Partir a pedra grande dá 2 pontos e partir uma pequena dá 1 ponto. Cada autobulldozer tem uma pontuação própria. No entanto, os autobulldozers funcionam como uma equipa pelo que deve ser também mantido o somatório dos pontos obtidos pelos autobulldozers (sugestão: use uma variável `static` na classe do `AutoBulldozer`).

7) Acrescente no `StoneBreaker` (e na classe `Bulldozer`) o código necessário para se conseguir visualizar quantos pontos tem o jogador e quantos pontos tem a equipa dos autobulldozers – utilize o método `setStatusMessage(…)` da classe `ImageMatrixGUI`. [Opcional] Ao terminar o jogo poderá aparecer uma mensagem a dizer se o jogador ganhou (ou não) à equipa dos autobulldozers.