Universidade de Coimbra

Faculdade de Ciências e Tecnologia



**Bejeweled**

Autores

Kevin Soares Duarte - 2011159671

Eugeniu Prozorovschii - 2011159654

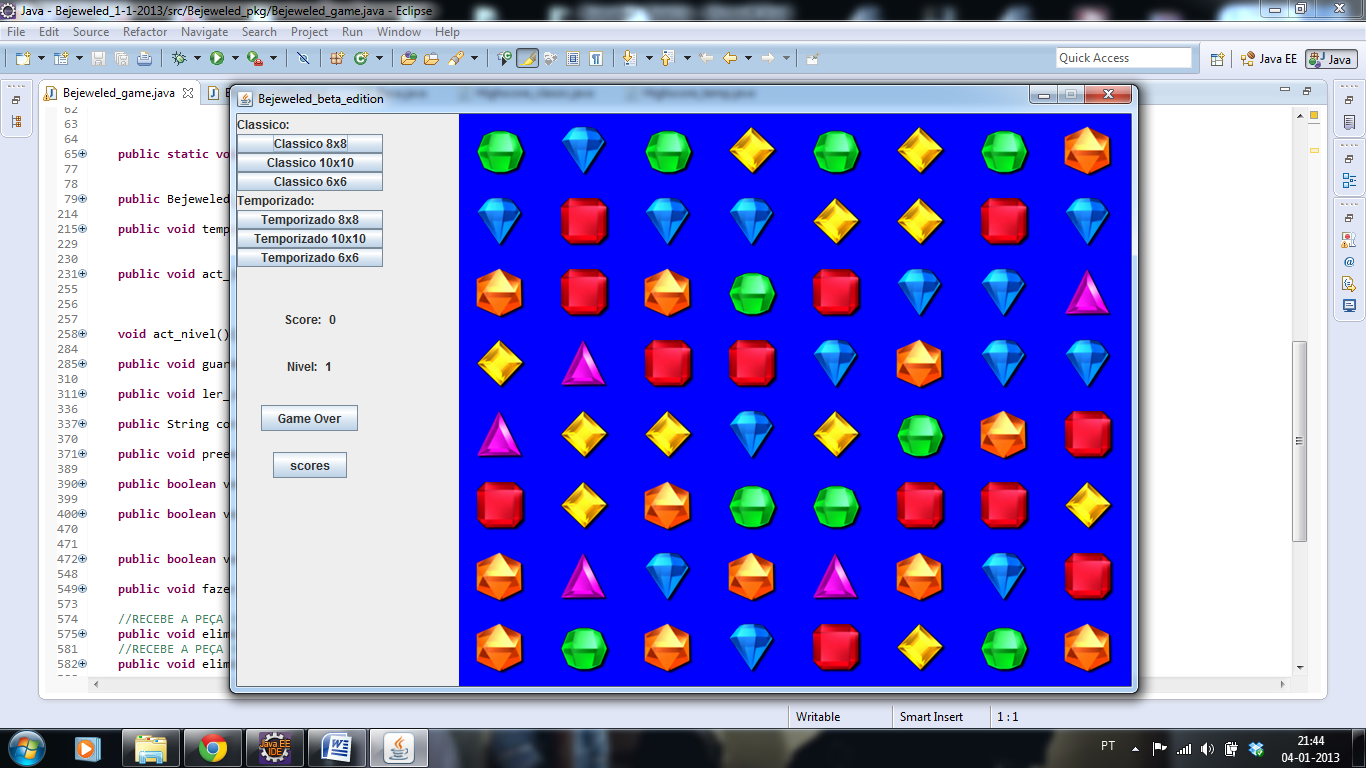
No âmbito da cadeira de Programação Orientada a Objetos

# Introdução

O objetivo do trabalho é desenvolver um jogo do mesmo género que o "Bejeweled", utilizando a linguagem de programação "Java". Neste trabalho serão aplicadas técnicas de programação que nos foram ensinadas ao longo do semestre.

# O jogo

O jogo é constituído por um tabuleiro de jogo com várias peças diferentes (6 tipos). Também será possível alterar o tamanho do tabuleiro, conforme o jogador desejar, tendo 3 opções (6x6, 8x8, 10x10). Existem também dois modos de jogo, o clássico e o temporizado. No modo de jogo clássico o jogador tem como objetivo alcançar o maior número de pontos até não existirem mais combinações possíveis, sendo que também é possível interromper o jogo clicando no botão "Game Over". No modo temporizado, o jogador tem dois minutos para cada nível, e cada nível tem um objetivo (número de pontos a alcançar para passar de nível), se o tempo acabar sem que o objetivo seja alcançado, o jogo acaba. O jogo consiste em formar uma linha ou coluna de 3 ou mais peças iguais para estas serem eliminadas, devolvendo pontos ao jogador, que posteriormente serão guardados num ficheiro por ordem decrescente (do jogador que fez mais pontos ao jogador com menor pontuação). Existem também tipos de peças chamadas de "Power-Up", que tem um poder especial.



# Classes

Classe "**Gere\_eventos**" - Esta classe tem como objetivo atribuir um evento a um determinado botão quando este é pressionado. Como por exemplo, quando clicamos no botão "Clássico 8x8" é iniciado o modo de jogo clássico com o tamanho do tabuleiro 8 por 8. Se pressionarmos o botão "Temporizado 8x8", é iniciado o modo de jogo com temporizador e com o tabuleiro de tamanho 8 por 8. O botão "Game Over" faz com que o jogo acabe, e o botão "Ver Scores" tem como função mostrar uma lista de scores guardados. Cada peça em si também é um botão, então também é feita aqui a verificação para saber em que peça se clicou.

Classe "**Peca**" - Nesta classe é atribuído a cada peça uma cor e uma imagem (para serem identificadas como peças distintas), um coordenada "x" e "y" para terem uma posição na matriz (no tabuleiro), e um poder, que inicialmente é inicializado a "zero", porque cada peça só vai ter um poder ("Power-Up") conforme a destruição de peças. O poder pode passar a "Um", a "Dois" ou a "Três" dependendo do tipo de poder.

Classe "**Highscore**" - Esta classe tem como atributos "nome", "score" e "nível". Quando o jogo acaba, o jogador insere o seu nome, e vai ser guardado num ficheiro o nome do jogador com os pontos obtidos e o nível alcançado. A classe "Highscore" terá uma subclasse "Highscore\_tempo" que vai herdar o "nome", o "score" e o "nível" da classe "pai" e também acrescentará tempo total utilizado durante o jogo.

Classe "**Bejeweled\_game**" - É a classe principal que contém o "main" e todas as funções necessárias para o programa correr corretamente.

# Funções

Neste tópico iremos apresentar todas as funções usadas, dando uma breve explicação sobre o funcionamento delas.

* **temporizador()** - é a função que atribui o tempo a cada nível do jogo no modo temporizado.
* **constroi\_tab()** - constrói o tabuleiro de jogo
* **Bejeweled\_game()** - tem como função em carregar as imagens, criar as frames, os panels, os botões do menu, a progress bar, as labels, adicionar as imagens para um array.
* **cores\_numeros()** - devolve uma string referente ao índice do arraylist da sua imagem.
* **preenche\_inicial()** - preenche o tabuleiro com peças, aleatoriamente.
* **verificar\_jogada()** - verifica se as peças que queremos trocar estão encostadas, não deixando trocar peças que não estejam uma ao lado da outra.
* **verificar\_coluna()** - verifica se há 3 ou mais peças do mesmo tipo na mesma coluna.
* **verificar\_linha()** - verifica se há 3 ou mais peças do mesmo tipo na mesma linha.
* **fazer\_poderes()** - executa o Power-Up, ou seja, se uma peça for considerada um Power-Up é esta a função que vai dar a ordem para o Power-Up utilizar o seu poder.
* **eliminar\_linha()** - função que faz com que as peças desapareçam do tabuleiro se for criada uma linha com 3 ou mais peças do mesmo tipo.
* **eliminar\_coluna()** - o mesmo que a função acima, mas para colunas.
* **descer\_pecas()** - função que simula a gravidade, ou seja, faz com que as peças caiam para os espaços vazios.
* **preencher\_vazio()** - preenche os espaços vazios com novas peças aleatoriamente.
* **somar\_score()** - incrementa uma variável com os pontos, conforme a destruição de peças.
* **actualizar\_matriz()** - atualiza as imagens das peças quando elas são trocadas.
* **verificar\_inicio()** - quando o jogo é iniciado, esta função verifica se não existem 3 ou mais peças iguais na mesma linha ou coluna, para o jogo não começar logo com pontos.
* **act\_nivel()** - função que serve para fazer com que o nível do jogo aumente, conforme o jogador ganhe os pontos. Cada nível tem um objetivo.
* **guardar score()** - função que regista num ficheiro o score obtido por um jogador, junto com o seu nome, o nível em que estava quando acabou o jogo, e no modo temporizado também regista o tempo gasto ao longo do jogo.
* **ler\_score()** - lê do ficheiro e mostra numa janela os scores guardados pelos jogadores.