

Programação III

Semestre de Inverno de 2022-2023

3º Trabalho prático

Data de Entrega: 29 de janeiro de 2023

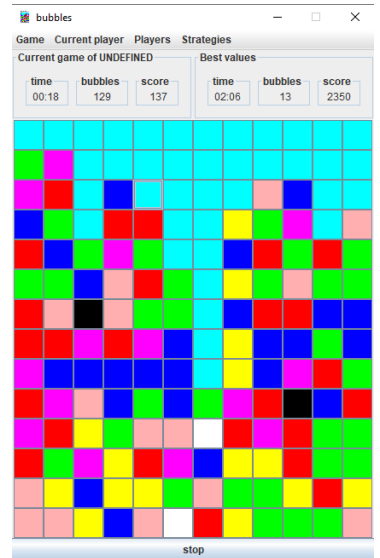
Completar o jogo *bubbles*.

O jogo consiste num tabuleiro preenchido com várias bolhas coloridas. O objetivo do jogo é destruir o maior número de bolhas para fazer o maior número de pontos no mínimo tempo. Para destruir as bolhas primeiro é necessário selecionar um conjunto de pelo menos duas bolhas.

Quando é feito um *click* no botão esquerdo do *mouse* numa determinada bolha é selecionado um conjunto de bolhas, o número de bolhas selecionadas depende da cor da bolha:

- Se a bolha for branca seleciona todas as bolhas da linha.
- Se a bolha for preta seleciona todas as bolhas da coluna.
- Se a bolha for vermelha, verde ou azul seleciona as bolhas da mesma cor que se encontram em cruz (esquerda, direita, cima e baixo).
- Se a bolha for amarela, rosa ou magenta seleciona as bolhas da mesma cor que se encontram a toda a volta (além de esquerda, direita, cima e baixo também seleciona as da diagonal).

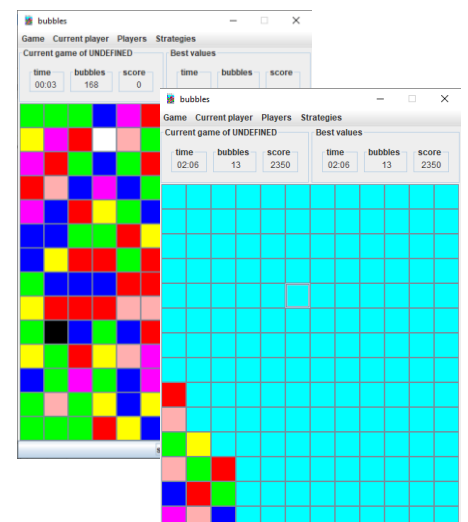
Quando é feito um *click* no botão esquerdo do *mouse* numa das bolhas do conjunto selecionado todas as bolhas do conjunto são destruídas e as que se encontram por cima descem (estratégia denominada gravitacional). Caso uma coluna fique vazia as que se encontram à direita são deslocadas para a esquerda.



A pontuação p ao longo do jogo vai sendo atualizada dependendo da cor c e do número n de bolhas de cada conjunto removido: $p = \sum c \times 2^{n-2}$. Quando o jogo termina a pontuação tem que refletir o tempo que o jogo demorou e o número de bolhas que não foram destruídas.

Usando as classes disponibilizadas em anexo como base implementar o que falta no jogo:

1. Implementar:
 - Classe **Black** que permite selecionar toda uma coluna.
 - Classe **DiagonalBubble** que permite selecionar todas as bolhas da mesma cor em cruz ou na diagonal.
 - Os métodos e escrever os comentários em falta nas classes **BubbleGame** e **StrategyGravitational**.
2. Acrescentar à janela o botão *start/stop*, inicialmente o botão tem na face “*start*”, após ser premido fica com “*stop*”. Quando é premido o “*stop*” o botão fica novamente com “*start*” na face.
3. Acrescentar à janela informação sobre:
 - O jogo corrente: tempo, número de bolhas e pontuação. O tempo e a pontuação começam a zero, o número de bolhas começa com o de bolhas coloridas que se encontram no tabuleiro. O tempo deve ser atualizado em cada segundo e o número de bolhas e a pontuação que são destruídas bolhas
 - Os melhores valores: menor número de bolhas e o respetivo menor melhor pontuação. Os melhores valores devem ser atualizados sempre termina um jogo



4. Acrescentar ao menu “*Game*”:


- Item “*How to play*” para escrever numa janela de diálogo uma pequena descrição do jogo e as suas regras. A descrição e as regras constam no ficheiro de texto “bubbles.txt”. O ficheiro “bubbles.txt” não é fornecido pelo que têm que o criar.

5. Acrescentar ao *menu bar* o menu “*Current player*” para apresentar informação sobre o jogador corrente, ou para mudar de jogador:

- Item “*new player*” - Mudar o jogador (não podem existir dois jogadores com o mesmo nome). A informação sobre os melhores valores têm que ser atualizados, caso já exista em memória informação sobre o jogador.
- Item “*statistics*” - Para mostrar as estatísticas do jogador. Número de jogos: total e número de jogos ganhos. Só é considerado um jogo ganho se o jogador terminar o jogo com um número de bolhas inferior a 2/3 do total. Pontuação: maior número de pontos e média dos pontos. Tempo: o melhor tempo para o menor número de bolhas e a média de tempos. Número de bolhas: mínimo, média e máximo de bolhas que não são destruídas.
- Item “*clear*” - Colocar a zero todos os parâmetros das estatísticas do jogador corrente.

Deve ser criada uma estrutura de dados que armazene os dados necessários para responder a todos os itens da estatística e outra para armazenar em memória as estatísticas de cada um dos jogadores.

6. Acrescentar ao *menu bar* o menu “*Players*” para;

- item “*save*” - Guardar em ficheiro as estatísticas de todos os jogadores que estão em memória. Quando a aplicação termina (pelo botão de fecho da janela  ou pelo item “*exit*” do menu “*Game*” se o jogador o pretender também devem ser guardadas as estatísticas dos jogadores.
- Item “*load*” – Ler as estatísticas dos jogadores que se encontram num ficheiro para a estrutura de dados em memória. Caso um jogador conste no ficheiro e em memória deve ser perguntado se é para acumular ou substituir.
- Item “*top 10*” – Mostrar os 10 jogadores que têm maior pontuação. Para a mesma pontuação por número de bolhas, para o mesmo número de bolhas por tempo.

7. (Opcional) Acrescentar ao *menu bar* o menu “*Strategy*” para

- Item “*gravitational*” – As bolhas quando são destruídas caem na vertical.
- Item “*shifter*” – As bolhas quando são destruídas depois de caírem na vertical são deslocadas para a esquerda. Defina uma classe que implemente **Strategy** para realizar esta estratégia.

Entrega do trabalho

O relatório deve conter:

- A descrição como interagem os componentes principais do jogo (**BubbleGameFrame**, **BubbleGame**, **Strategy**) e a descrição dos métodos mais relevantes.
- A descrição de como funcionam os *listeners* do jogo (**GameListener** e **BubbleListener**). Na descrição deve constar: quando são adicionados e a quem são adicionados; quando são chamados e quem os chama.
- Para cada um dos elementos adicionados ou modificados deve ser explicado:
 - O local onde foram colocados na hierarquia de classes fornecida.
 - Os membros que foram acrescentados.
 - Os métodos mais relevantes.
- Uma descrição das estruturas de dados implementada para armazenar as estatísticas e quem as atualiza. Na descrição deve constar:
 - Os dados.
 - Os métodos mais relevantes.
- Diagramas estáticos de classes UML (incluindo as que já estão implementadas).
- Um capítulo onde se evidencie quais dos temas estudados ao longo do semestre foram aplicados e quais as vantagens resultantes da sua aplicação na implementação do jogo. Em particular, os seguintes temas: herança e polimorfismo; estruturas de dados; interfaces funcionais; *streams*; e programação *event driven*.

Na conceção das soluções note que se valoriza:

- A qualidade do código produzido (facilidade de alteração; não repetição de código; modularização; documentação);
- As estruturas de dados e os algoritmos utilizados;
- A facilidade de utilização da aplicação;
- Criatividade.

Bom trabalho