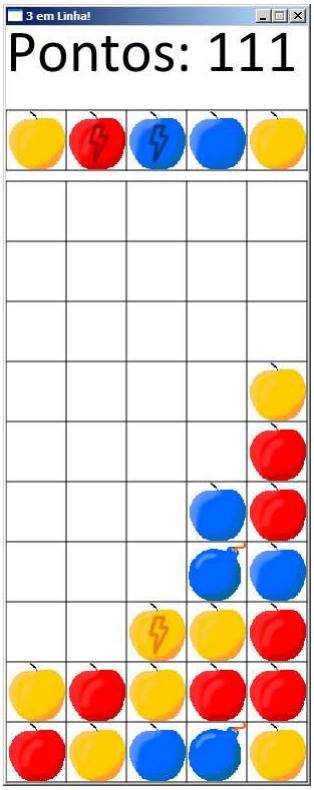
IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Programação Orientada por Objectos

Projecto: cursos EEC – EI - EB

3 em linha!



Ano Lectivo: 2011/2012

Época Normal e de Recurso

Índice

Introdução

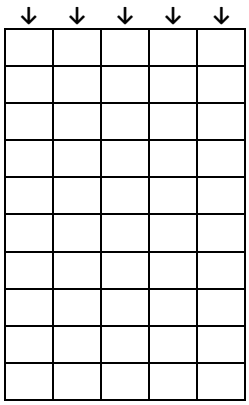
O objectivo deste projecto é desenvolver, utilizando a linguagem Java e a metodologia de Programação Orientada por Objectos, um programa que será uma implementação de uma versão de jogos do tipo "3 em linha". O jogo a implementar será uma versão simplificada do jogo Bad Apples disponível para iOS (http://www.metaversalstudios.com/games/badapples/).

Durante o desenvolvimento do presente projecto pretende-se que o aluno consiga demonstrar os seus conhecimentos na área da POO e não que implemente um jogo real. A interface gráfica subjacente a este jogo, apesar de bastante simples (apenas temos um reticulado de botões), permite a criação de ecrãs bastante coloridos e, caso o aluno deseje, poderá ao aprofundar a utilização das animações disponíveis na tecnologia JavaFX, conseguir um resultado bastante apelativo.

O Jogo

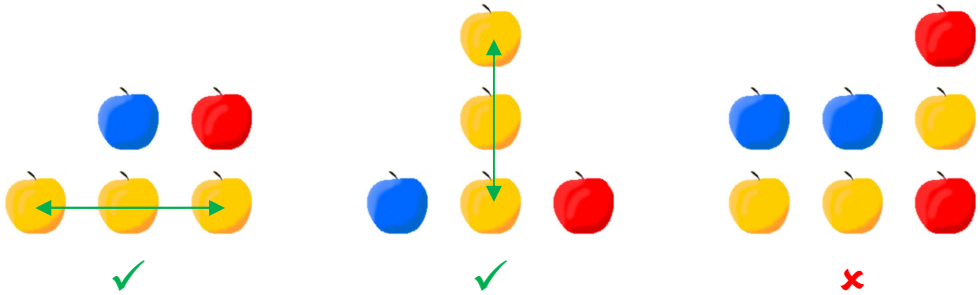
Regras básicas

O jogo é baseado num tabuleiro constituído por um conjunto de linhas (10 por omissão) e colunas (5 por omissão), onde são colocadas, pelo computador, peças a partir do topo:

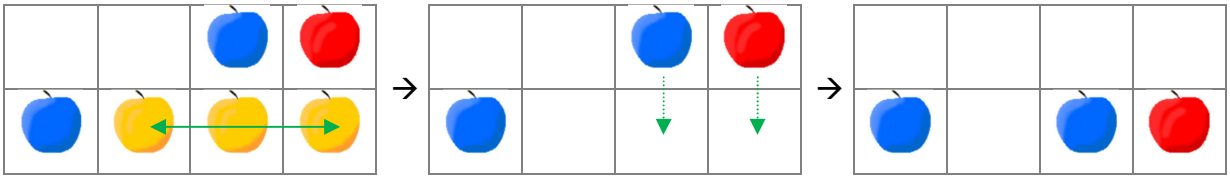


Existem 3 famílias de peças com cores distintas: **Amarela**, **Azul** e **Vermelho**, que poderão ser representadas por simples quadrados, com as cores respectivas, ou por ícones mais apelativos:

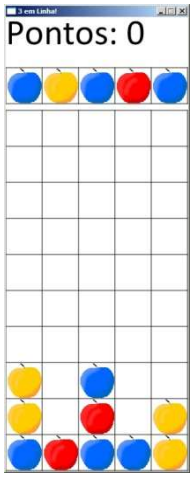
Pretende-se colocar, pelo menos 3, peças da mesma cor em sequência (em linha ou coluna):



Quando existe uma sequência de 3, ou mais peças, da mesma cor, estas são removidas do tabuleiro (contando um ponto por cada peça) e as peças que se encontram em cima destas, são deslocadas, automaticamente, para ocupar as casas vazias:



O jogo evolui jogando o computador e o utilizador à vez: quando o computador joga introduz uma nova peça em cada coluna; quando o utilizador joga move, sempre, UMA peça, tentando criar uma sequência ou preparando caminho para o conseguir numa jogada subsequente (no topo do tabuleiro estão indicadas as peças que serão introduzidas na próxima jogada do computador):



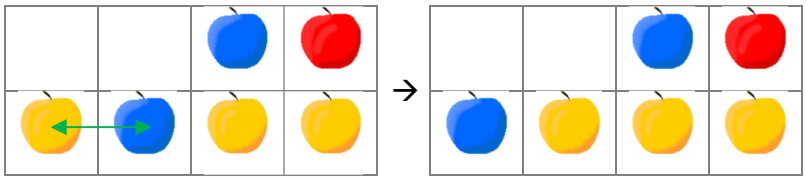
←Pontuação atingida (nº peças eliminadas)

←Próximas peças a introduzir pelo computador

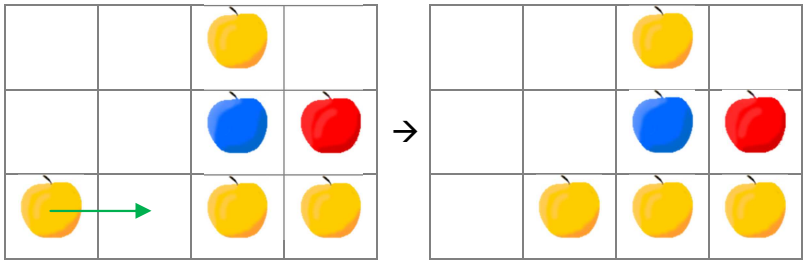
←Tabuleiro com as peças em jogo

Após cada jogada são retiradas as eventuais peças que façam sequência, de 3 ou mais da mesma cor, antes de o oponente realizar a sua jogada. Se com a movimentação das peças devido às casas deixadas vazias se obtiverem novas sequências o processo repete-se: removem-se essas novas peças em sequência e deslocam-se as que se encontram acima delas.

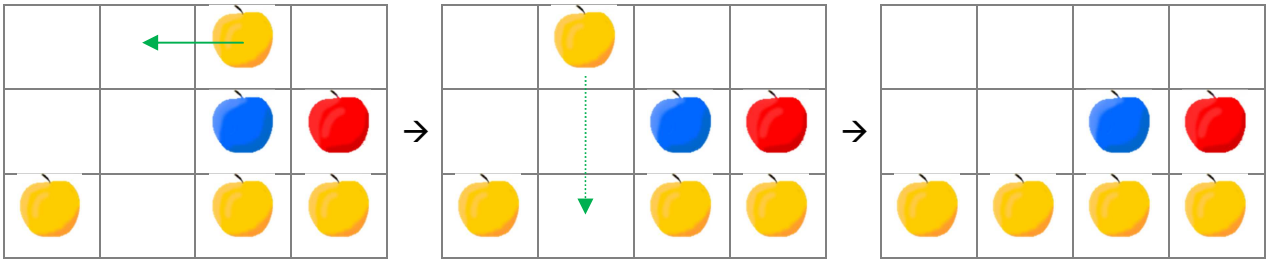
Na vez do utilizador este deve SEMPRE mexer uma peça. O movimento poderá ser trocar duas peças de posição:



Ou mover uma peça para uma zona vazia:

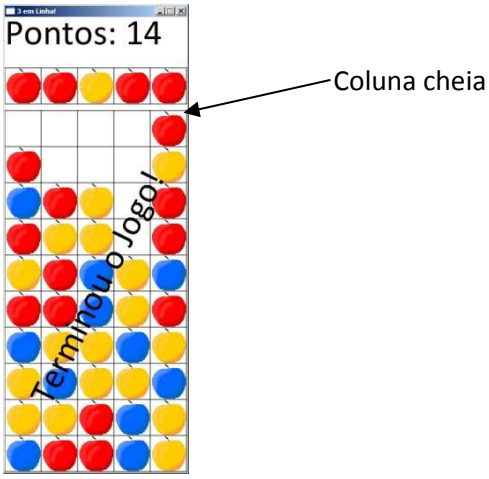


Caso a movimentação da peça, por parte do utilizador, a coloque numa posição sem peças abaixo, esta será deslocada, automaticamente, passando por todas as linhas vazias, até uma posição com peças abaixo ou ficando na base do tabuleiro:



Se, após a jogada do computador (introdução de uma nova peça em cada coluna) não existir nenhuma peça no tabuleiro para o utilizador mover (porque foram todas removidas devido a terem sido efectuadas sequências), este perde a vez e o computador volta a introduzir um conjunto de novas peças.

O jogo termina quando, pelo menos, uma coluna fica completamente preenchida, não sendo possível, ao computador, a introdução de novas peças:



Para não começar com um tabuleiro vazio, o jogo inícia-se com o computador a introduzir um número de peças (10 por omissão, equivalente a duas linhas), com cores escolhidas aleatoriamente em colunas aleatórias.

Peças especiais

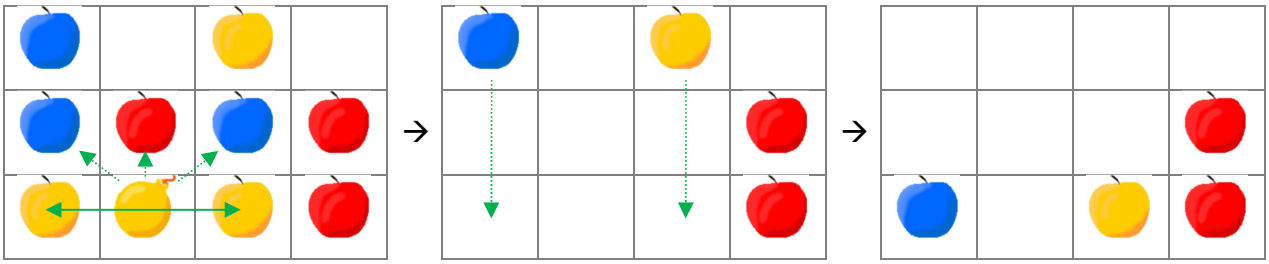
Para tornar o jogo mais diversificado e para poder demonstrar melhor os seus conhecimentos em POO. São acrescentados três tipos de peças novas (mantendo no entanto a sua cor base), cada um com um efeito especial:

- Peça explosiva
- Peça raio
- Peça muda de cor

Peça explosiva

Uma peça "explosiva" deve ter uma representação que a permita distinguir-se das restantes (por exemplo no caso da peça amarela):

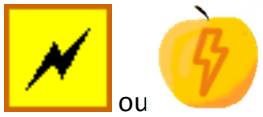
Quando uma peça "explosiva" se encontra numa sequência de 3 ou mais peças da mesma cor e vai ser retirada, ela "explode" retirando todas as peças imediatamente adjacentes que, eventualmente, não seriam removidas numa situação normal:



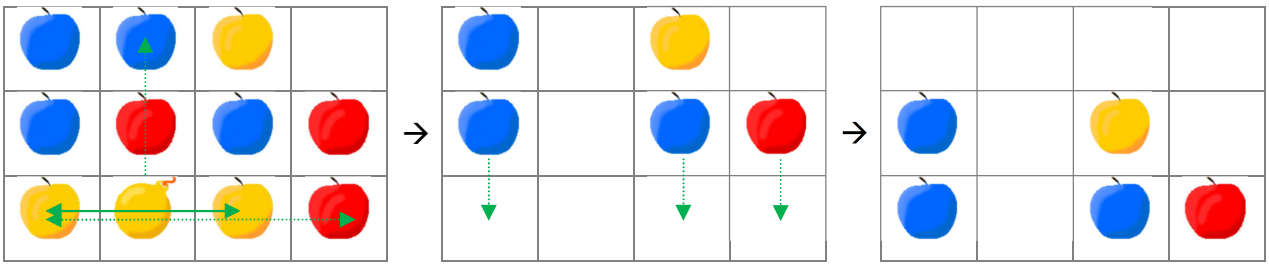
Se de entre estas novas peças afectadas, que irão ser removidas, existir uma peça especial (explosiva ou com outro dos efeitos explicados em seguida) o seu efeito é "activado", repetindo-se o processo.

Peça raio

Uma peça "raio" deve ter uma representação que a permita distinguir-se das restantes (por exemplo no caso da peça amarela):



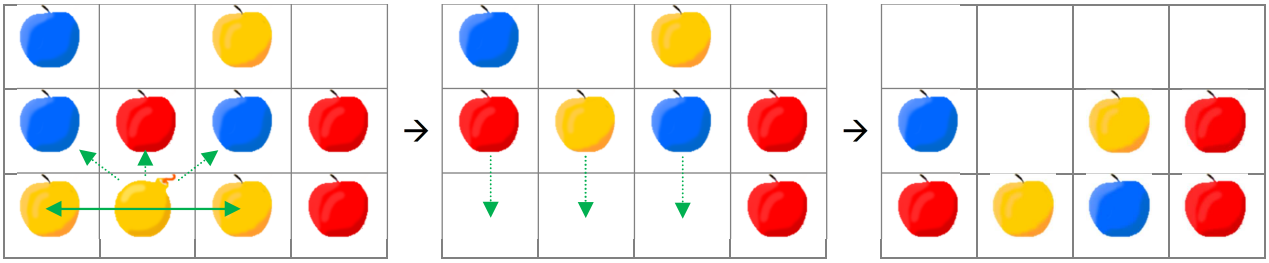
Quando uma peça "raio" se encontram numa sequência de 3 ou mais peças da mesma cor e vai ser retirada, ela "emite um raio" que retira todas as peças que se encontram na mesma linha e na mesma coluna:



Peça muda de cor

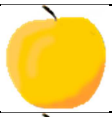
Uma peça "muda de cor" deve ter uma representação que a permita distinguir-se das restantes (por exemplo no caso da peça amarela):

Quando uma peça "muda de cor" se encontram numa sequência de 3 ou mais peças da mesma cor e vai ser retirada, ela afecta as peças imediatamente adjacentes e de cores diferentes que não serão removidas alterando as suas cores para outra (eventualmente a mesma) aleatoriamente:



Resumo

Existem no total 12 peças distintas, a que corresponde 3 famílias, uma de cada cor com, eventualmente, cada um dos 3 efeitos especiais:

Cor \ Efeito	Simples	Explosiva	Raio	Muda de Cor
Amarela				
Azul				
Vermelha				

Existência de 3 ou mais peças da mesma cor em sequência (numa linha ou coluna) implicam na sua remoção, sendo as casas deixadas vagas ocupadas pelas, eventuais, peças existentes acima. Se durante esta reorganização se criarem novas sequências de 3 ou mais peças da mesma cor o processo recomeça (são removidas as peças e reorganizadas as acima). Quando uma peça especial é removida ela aplica um efeito, explosão, raio ou mudar de cor, que afectam as peças adjacentes ou na mesma linha/coluna.

Sugestões de Desenvolvimento

Comece por implementar o jogo assumindo que não há peças especiais. Estas serão acrescentadas, por especialização, após toda a funcionalidade estar testada.

Implemente, na primeira fase, o jogo recorrendo a uma interface de consola onde cada cor será representada através de uma letra: **a**-marela; az-**u**-l e v-**e**-rmelha:

```
   eaaau
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04    a  04
03    a  03
02 aa e  02
01 auaua 01
   12345
Pontos: 0
```

E a indicação da jogada do utilizador será dada através da indicação da linha e coluna:

```
Indique a posição da peça a mover[10,5]: 3 4
Indique a posição da peça destino[10,5]: 2 3
```

Sendo apresentado um "tabuleiro" por cada etapa:

```
   eaaau
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04       04
03    a  03
02 aaae  02
01 auaua 01
   12345
Pontos: 0

   eaaau
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04       04
03    a  03
02    e  02
01 auaua 01
   12345
Pontos: 3
```

```
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04    a  04
03    a  03
02 eaaeu 02
01 auaua 01
   12345
Pontos: 3

   uauau
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04    a  04
03    a  03
02 eaaeu 02
01 auaua 01
   12345
Pontos: 3

Indique a posição da peça a mover[10,5]:
```

Na implementação do programa, comece por escolher e implementar as classes necessárias ao armazenamento da informação (tabuleiro, próximas peças, etc.).

Tal como sugerido, implemente as regras do jogo assumindo que não existem peças especiais. Escreva métodos para cada uma das etapas do jogo:

1. Jogada do Utilizador (indicar peça a mover: origem e destino)
2. Retirar eventuais peças em sequência
3. Compactar as colunas com espaços vazios
4. Voltar à etapa 2 até não haver peças em sequência
5. Caso haja uma coluna cheia o jogo termina
6. Adicionar as próximas peças do computador
7. Retirar eventuais peças em sequência
8. Compactar as colunas com espaços vazios
9. Voltar à etapa 6 até não haver peças em sequência
10. Gerar próximas peças a serem introduzidas pelo computador no futuro
11. Voltar à etapa 1 para o jogar poder recomeçar

Para retirar as peças que estejam em sequência da mesma cor (3 ou mais peças), será necessário percorrer todas as posições e determinar se a peça está em sequência na horizontal e/ou vertical com outras da mesma cor. No entanto, não se deve remover logo todas essas peças pois pode acontecer que uma peça que esteja em sequência com a actual vá também estar em sequência com outras na sua vizinhança, o que só será detetado quando o teste “lá chegar”. Assim será conveniente marcar as peças como “a remover” e só no final de fazer a passagem completa por todo o tabuleiro, retirar, de facto, essas peças.

Quando todo funcionamento estiver garantido pode introduzir as peças especiais. Como sugestão utilize na representação um acento para distringuir os diferentes tipos de peças especiais:

Cor \ Efeito	Simples	Explosiva	Raio	Muda de Cor
Amarela	a	à	á	â
Azul	u	ù	ú	û
Vermelha	e	è	é	ê

```
   euaeu
   12345
10       10
09       09
08       08
07       07
06       06
05       05
04 a     04
03 a u   03
02 eua   02
01 ûaea  01
   12345
Pontos: 0
```

Para a implementação da interface gráfica crie 12 ícones diferentes que serão associados a cada tipo de peça (cor + efeito).

A melhor forma de apresentar etapas na interface gráfica consiste em implementar *Animations* onde no evento OnFinished será iniciada a *Animation* da etapa seguinte (será explicado nas aulas teóricas).

Regras Gerais de Implementação

- O programa deve ser desenvolvido utilizando a linguagem Java e os métodos de programação orientada por objectos.
- Para os identificadores siga as convenções adoptadas normalmente para a linguagem Java:
  - A notação camelCase para o nome das variáveis locais e identificadores de atributos e métodos;
  - A notação PascalCase para os nomes das classes e interfaces;
  - Maiúsculas para os nomes das constantes e dos valores enumerados;
  - Não utilize o simbolo '_' nos identificadores (excepto nas constantes), nem abreviaturas.
- Não existem restrições quanto às características e/ou funções da linguagem Java utilizadas, sendo, no entanto, requerido que o programa seja desenvolvido num ambiente gráfico usando o JavaFX.
- Toda a interface com o utilizador fica ao critério do programador, sendo premiadas a facilidade de utilização e a apresentação.
- É necessário que o projecto cumpra o que é pedido no seu enunciado, sendo deixado ao critério do programador qualquer pormenor de implementação que não seja referido no mesmo e que deve ser devidamente documentado.

Regras de Desenvolvimento e Entrega do Projecto

- Cada projecto deverá ser elaborado em **grupos de dois alunos**, podendo eventualmente ser elaborado individualmente. Não serão permitidos em nenhum caso grupos de mais do que dois alunos, nem grupos com alunos que pertençam a turmas de laboratórios com diferentes docentes.
- O projecto será entregue em duas fases:
  - Uma primeira fase (**até às 23:59:00 do dia 27/05/2012**) com uma interface em modo consola mas com todas as funcionalidades implementadas;
  - A segunda fase (**até às 23:59:00 do dia 17/06/2012**) consiste na implementação da interface gráfica, através de JavaFX, do sistema desenvolvido na primeira fase.
- O projecto deve ser entregue até à data limite especificada por **via exclusivamente electrónica utilizando a área dos trabalhos do respectivo Docente de Laboratório no Moodle 2.0**. Todos os ficheiros que compõem o projecto deverão estar guardados num único ficheiro compactado em **formato ZIP ou RAR**.
- Todos os materiais do projecto devem ser devidamente identificados com nome, número, turma, curso e endereço de email dos alunos. Incluir também o nome do professor de laboratório encarregado da avaliação nos relatórios ou manuais.
- **Projecto Final** – O projecto final deve incluir:
  - Um **Manual Técnico** onde conste uma breve descrição do programa, incluindo:
    - A descrição (objectivo, atributos e métodos disponíveis) das classes desenvolvidas e a sua interligação. Para as classes de ambas as fases.
    - A documentação do programa em **JavaDoc** (não converta o documento gerado automaticamente em HTML para .doc!)
  - O **código fonte do programa** na forma de projecto em NetBeans.

Regras de avaliação do Projecto

- A entrega do projecto depois do prazo estipulado implica a penalização de um valor por dia de atraso até um máximo de três valores. Depois disso o projecto não será aceite.
- A classificação do programa terá em conta a qualidade da programação (factores de qualidade do software), a estrutura do código criado segundo os princípios da programação orientada por objectos, tendo em conta conceitos como a coesão de classes e métodos, o grau de acoplamento entre classes e o desenho de classes orientado pela responsabilidade, e a utilização/conhecimento da linguagem Java.
- Serão premiadas **a facilidade de utilização, a apresentação, a imaginação e a criatividade**.
- O projecto terá uma componente de avaliação oral obrigatória com classificação individual dos elementos do grupo. Os alunos que não comparecerem à discussão serão classificados com zero. Nesta discussão com os alunos será apurada a capacidade do aluno de produzir o código apresentado. Nos casos em que essa capacidade não for demonstrada a nota atribuída será zero.
- A avaliação oral é realizada pelo respectivo professor de laboratório e irá ser feita uma marcação prévia para cada grupo de trabalho.

- Todos os projectos serão submetidos a um sistema de detecção de cópias automático. Os projectos que forem identificados como cópias e que se verificar serem **cópias serão anulados**.

Critérios de Avaliação

O projecto será avaliado quantitativamente segundo os seguintes critérios:

Funcionalidades	**60%**
Interface gráfica	**25%**
Elementos gráficos e layout (posicionamento dos botões)	5%
Funcionalidade disponibilizadas (definições dos eventos)	5%
Animações	15%
Funcionalidades Específicas	**35%**
Geração aleatória das próximas peças a entrar em jogo	5%
Introdução das peças no tabuleiro	5%
Compactação das colunas quando há peças retiradas ou foi movida uma peça do utilizador	10%
Detecção de 3 ou mais peças em linha e remoção das mesmas	15%
Implementação	**30%**
Definição de herança (ex: utilização nas classes das Peças)	10%
Utilização do polimorfismo (ex: utilização no Retirar das Peças)	5%
Utilização de constantes (sua definição e tornar o sistema dependente destes valores)	5%
Definição das Classes (ex: para armazenar a informação do tabuleiro e próximas peças)	10%
Manual Técnico	**10%**
Descrição do Programa	5%
Documentação JavaDoc	5%
Penalizações	**-10%**
Erros de código (bugs)	5%
Problemas de coesão e acoplamento, legibilidade do código (identificadores)	5%
Bonificações	**+10%**
Utilização de componentes e caraterísticas não lecionadas	5%
Imaginação e criatividade (funcionalidades úteis que não foram pedidas)	5%