Aula Prática 8 - Imagens a cores

Nesta aula são propostos exercícios que envolvem imagens a cores através da classe ColorImage, que por sua vez utiliza objetos Color. Para a resolução dos exercícios desta semana e que servirão de base ao Projeto Individual iremos utilizar algumas classes para ajudar na manipulação de imagens. Descarregue os ficheiros anexados (Color.java, ColorImage.java, e ImageUtil.java) e copie os mesmos para a pasta *src* do projeto do Eclipse.

Trabalho prévio

Construa uma classe estática TestClass, de modo a permitir:

1. Criar um objecto do tipo String e visualizar no pandionJ (introduza um *breakpoint*):

```
class TestClass {
	static void test () {
		String nullString = null;
		String emptyString = "";
		String s1 = "Olá mundo";
	...
}
```

2. Acrescente as seguintes variáveis, e experimente a função estática valueOf () da classe String:

```
	...
	String s4 = String.valueOf (42);
	String s5 = String.valueOf ('Z');
	String s6 = String.valueOf (3.14);
	String s7 = String.valueOf (true);
```

3. Teste os métodos de instância length() e charAt() da classe String:

```
...
int length = s1.length();
char c = s1.charAt (2);
```

4. Converta String em char [ ] e vice-versa:

```
...
char [] word = s1.toCharArray();
String s3 = new String (word);
```

5. Utilize a função drawText (int textX, int textY, String text, int textSize, Color textColor):

```
...
ColorImage img = new ColorImage(200, 150);
img.drawText(30, 50, "And the answer is ..", 20, new Color(255,255,255));
img.drawCenteredText(100, 100, s4, 20, new Color(255,255,255));
```

Aula prática:

Abra agora a classe Color, e modifique-a, de modo a permitir:

1. Aceder a constantes que representam as cores mais comuns (branco, preto, vermelho, verde, azul, etc.). Exemplo:

```
class Color {
        static final Color RED = new Color(255, 0, 0);
        ...
}
```

2. Obter a cor inversa (255 - *R*, 255 - *G*, 255 - *B*).
3. Obter uma cor mais clara/escura, dependendo de um valor (positivo - mais clara; negativo - mais escura).

```
Color brighter(int value) { … }
```

4. Obter uma conversão para uma cor em tom de cinzento. O tom de cinzento correspondente a uma cor pode ser obtido através do valor da luminância da cor.
5. Saber se outra cor é igual, isto é, se os 3 valores RGB são iguais ao da cor passada no parâmetro.

   ```
   boolean isEqualTo(Color c) { … }
   ```

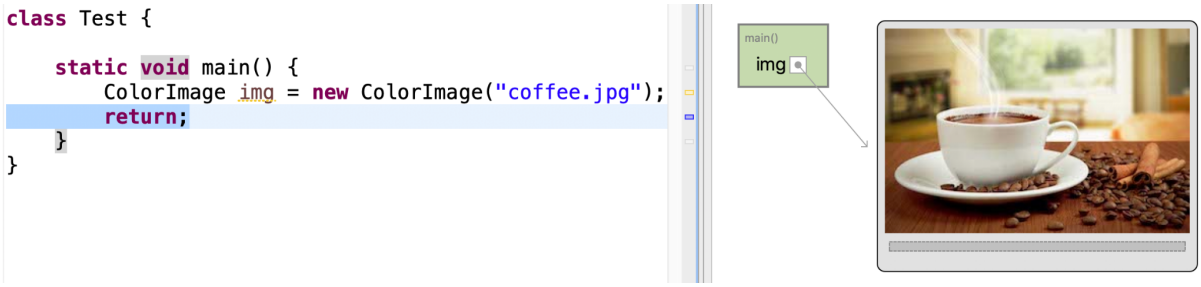
6. Saber se a cor está presente num vetor. Deverá ser utilizado o método desenvolvido na alínea anterior para verificar a igualdade entre cores.

   ```
   Color[] v = { Color.RED, new Color(220, 220, 220), Color.BLUE }

   Color c = new Color(220, 220, 220);

   boolean isInVetor = c.containedIn(v); // true

   boolean notInVetor = Color.GREEN.containedIn(v); // false
   ```

Imagens a cores

Para experimentar exercícios com imagens, é aconselhável utilizar imagens relativamente pequenas (pe. até 300x300 pixels), para que seja prático visualizar (dado o espaço disponível), e também para que o processo de execução passo a passo não se torne demasiado lento.

Para carregar uma imagem a cores de um ficheiro, construir um objeto ColorImage fornecendo o nome do ficheiro, tal como ilustrado na imagem abaixo. O ficheiro terá que estar localizado na **raiz** do projeto (não no diretório **src**), por forma a ser encontrado.



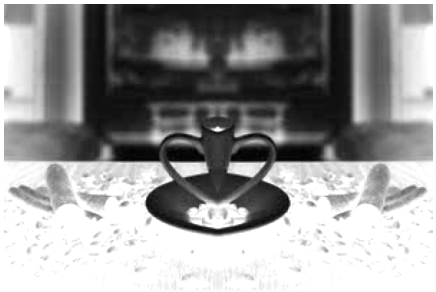
Observe que a classe ColorImage tem como atributo uma matriz de inteiros e não uma matriz de objetos, pois tal implementação torna-a mais eficiente. Nalgumas operações podemos manipular diretamente os valores da matriz, ao passo que noutras é mais adequado manipular os objetos Color.

1. Modifique a classe ColorImage, utilizando as funções do trabalho prévio quando apropriado, de modo a que existam operações para:
    a. Obter uma cópia da imagem.
    b. Transformar a imagem, invertendo as suas cores.

    c. Obter uma nova imagem a preto e branco (tons de cinzento).

    d. Transformar a imagem, tornando-a mais clara de acordo com um valor dado. Um valor positivo torna a imagem mais clara, um valor negativo torna a imagem mais escura.
    e. Transformar a imagem, de forma a ficar espelhada horizontalmente.

2. Desenvolva uma função **estática** que dada uma imagem, produza uma nova com o seguinte efeito. Será útil desenvolver um procedimento extra, para além dos realizados anteriormente.



3. Desenvolva um procedimento de ColorImage para colar (*paste*) outra imagem sobre, num dado ponto.

   class ColorImage { ...

   ```
   void paste(ColorImage img, int x, int y) { … }
   ```

   }

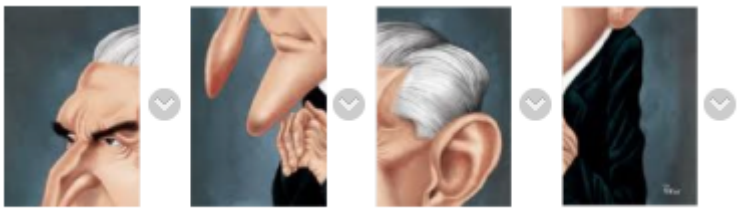
4. Problema: produzir a seguinte imagem (a partir da foto original). A solução será baseada apenas numa integração de funções e procedimentos realizados anteriormente.

Exercícios extra

1. Desenvolva uma função de ColorImage para obter uma seleção retangular da imagem, dados o canto superior esquerdo e o canto inferior direito.

   ```
   ColorImage selection(int startx, int starty, int endx, int endy) { ... }
   ```

2. Desenvolva uma função estática para criar um vector com quatro imagens, uma para cada quarto de uma imagem.



3. Construir uma imagem nova, juntando duas imagens horizontalmente.
   Exemplo:

   ```
   ColorImage colorimage3 = horizontalMerge (colorimage1, colorimage2);
   ```