IFBA – Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Ciência da Computação
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas
INF008 – Programação Orientada a Objetos
Prof.: Frederico Jorge Ribeiro Barboza – **Data:** 15/06/2021

Aluno: _____ Nota: _____

IIª Avaliação – 2021.1

Um sistema de avaliação de mapas para corrida de orientação funciona como se segue:

O sistema é composto por uma biblioteca de imagens, que representam mapas de corrida de orientação. Cada imagem é composta por uma coleção bidimensional de pixels. Como as imagens provém de fontes distintas de captação, eles podem ser representadas por sistemas de representação de cores distintos. Atualmente, o atlas pode armazenar dois tipos de mapas, os mapas RGB e os mapas CMYK, contudo estuda-se a possibilidade da incorporação de outros modelos de representação (mapas HSV, por exemplo). Os mapas RGB são compostos por coleções bidimensionais de pixels de cores no formato RGB, enquanto os mapas CMYK são compostos por coleções bidimensionais de pixels de cores no formato CMYK.

As cores RGB modelam um sistema de cores aditivas em que o Vermelho (**R**ed), o Verde (**G**reen) e o Azul (**B**lue) são combinados de várias formas de modo a reproduzir um largo espectro cromático. Uma cor RGB é representada por uma 3-upla, onde cada elemento pode assumir um valor de 0 a 255, representando a quantidade de vermelho, verde e azul que compõem a cor.

Já as cores CMYK modelam um sistema de cores subtrativas. Neste caso, cada cor é representada por uma 4-upla formado por ciano (**C**yan), magenta (**M**agenta), amarelo (**Y**ellow) e preto (Black (**K**ey)). Cada um dos elementos desta tupla pode assumir um valor entre 0 e 100.

Pelo fato de operar com vários padrões de cores distintas, o sistema verifica a similaridade de duas cores quaisquer, através da distância (módulo da diferença) entre suas luminosidades (tom de cinza). Portanto, toda cor tem uma luminosidade associada independente do padrão. A luminosidade do padrão RGB é calculada através da expressão luminosidade = (R*0.3 + G*0.59 + B *0.11), que deve ser truncada para um valor inteiro. A luminosidade de uma cor CMYK é dado pela quantidade de preto (K) na cor multiplicado por 255 divido por 100 (luminosidade = K * 255 / 100).

1) (3.0) – Escreva um modelo de classes que descreva adequadamente as classes de negócio do problema acima.

2) (4.0) – Escreva todos os métodos necessários para que o sistema receba uma 3-upla, representando uma cor RGB, um limiar de similaridade de luminosidade *l* e um percentual mínimo ***p*** e retorne todos os mapas que possuam ao menos o percentual mínimo ***p*** de pixels cuja luminosidade seja similar a da cor RGB informada em +/- ***l***% . Por exemplo, o sistema pode ter que retornar todas as imagens com ao menos 40% dos pixels com luminosidade de +/- 10% do azul puro (0, 0, 255).

```
public Imagem[] getImagemPorLuminosidade(double red, double green,
    double blue, double pctMinimo, double limiarSimilaridade);
```

3) (3.0) – O sistema quer incorporar a possibilidade de converter imagens de um sistema de cor para outro sistema de cor. Por exemplo, converter uma imagem em RGB para uma imagem CMYK; de uma imagem CMYK para uma imagem RGB, etc. O conversor deve criar uma imagem do tipo adequado (aquele que se

quer converter) dada as dimensões da imagem original, e depois acrescentar na posição correta um novo pixel representado na cor destino da conversão. Um conversor foi escrito, com esta finalidade e o código é fornecido abaixo.
Para que o conversor seja genérico ele foi baseado em interfaces. Adapte o conversor, para as classes do seu modelo e escreva o código da interface ConversorCor, e uma classe que possa ser utilizada pelo conversor para converter imagens de CMYK para RGB considerando as seguintes equações de conversão:

- R = 255 × (1-C)/100 × (1-K)/100
- G = 255 × (1-M)/100 × (1-K)/100
- B = 255 × (1-Y)/100 × (1-K)/100

```
public class Conversor {
      public Imagem converter(Imagem imagem, ConversorCor conversor){
            Imagem      novo      =      conversor.getNovoMapa(imagem.getAltura(),
imagem.getLargura());
            for(int iCont = 0; iCont < imagem.getAltura(); iCont++)
                  for(int jCont = 0; jCont < imagem.getLargura(); jCont++){
                        Cor cor = imagem.getPixel(iCont, jCont);
                        Cor novaCor = conversor.converter(cor);
                        novo.setPixel(iCont, jCont, novaCor);
                  }
            return novo;
      }
}
```