Universidade de São Paulo – Escola de Artes, Ciências e Humanidades Bacharelado em Sistemas de Informação Introdução à Ciência da Computação I Professoras Ariane e Fátima

EXERCÍCIO PROGRAMA 2 RECONHECIMENTO SIMULADO DE PLACAS DE VEÍCULOS

Neste Exercício Programa (EP) você vai aprender alguns conceitos de processamento de imagens, a fim de implementar, de forma simulada e simplificada, o reconhecimento dos dígitos de placas de veículo.

Processamento de imagens é uma área da Computação que visa desenvolver técnicas para manipular, armazenar e recuperar imagens considerando os mais diversos formatos e para as mais diversas finalidades. Entre as principais técnicas de processamento de imagens estão algumas para suavização, realce, reconhecimento de bordas, identificação de objetos e reconhecimento de padrões.

Uma imagem consiste em uma matriz (array bidimensional) de números inteiros com dimensão de $\mathbf{n} \times \mathbf{m}$, onde \mathbf{n} é a quantidade de linhas e \mathbf{m} é a quantidade de colunas. Os números representam cores dentro de um intervalo considerado. Cada número é chamado de \mathbf{pixel} . Em geral, as cores são níveis de cinza, sendo o valor zero relacionado à cor mais escura (preto) e o valor máximo da escala relacionado à cor mais clara (branca). Assim, um intervalo de 8 bits, permite armazenar os valores de 0 a 255. Quanto maior for o intervalo considerado, maior é a quantidade de nuances considerada na formação da imagem.

Em Java, há APIs (*Application Programming Interfaces*) que disponibilizam classes e métodos para manipulação de imagens. Uma imagem é fornecida em formato de vetor e, para facilitar o processamento, deve ser transformada em uma matriz bidimensional. Neste EP não vamos utilizar APIs. Um vetor deverá ser lido de um arquivo e convertido para uma matriz bidimensional. Daqui para frente chamaremos esta matriz de matrizimagem.

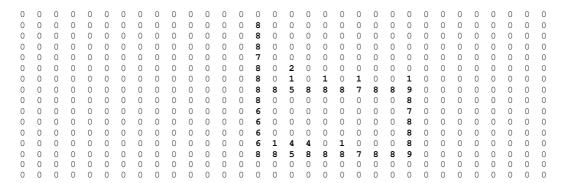
A partir desta introdução, vamos à apresentação do problema.

O objetivo é reconhecer de forma simplificada os quatro dígitos da placa de um veículo, realizando algumas operações sobre matrizes-imagens e, informar ao usuário quem é o proprietário do veículo (vamos assumir que as placas são compostas apenas por esses 4 dígitos, ignorando as letras). Os algarismos são formatos por traços horizontais e verticais e tem formatos fixos, conforme abaixo:



Vamos considerar que os objetos (números das placas com quatro algarismos) são números diferentes de zero e o fundo da matriz-imagem é composto por zeros. A identificação de um algarismo deve ser realizada com base nas suas bordas. No exemplo a seguir, a matriz-imagem representa o número 6.





A seguir são definidas as classes e alguns métodos obrigatórios. Você deve escolher o tipo de retorno e os tipos de parâmetros de forma adequada para cada método, assim como o tipo adequado para cada atributo das classes.

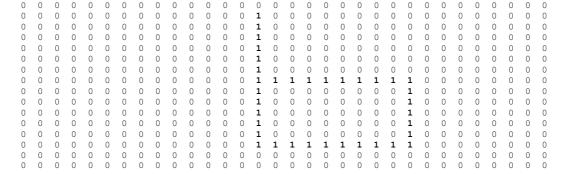
Atenção: seu programa será verificado obedecendo aos nomes das classes e métodos fornecidos. Portanto, observe atentamente as definições. Se necessário (e provavelmente será), crie outros métodos (adicionais) não definidos aqui.

Classe Imagem:

Esta classe contém métodos que executam operações sobre matriz-imagem.

A classe Imagem deve implementar pelo menos os seguintes métodos:

- converteVetorMatriz (vetorImg, nLinhas, nColunas): converte o array unidimensional vetorImg em uma matriz-imagem bidimensional com nLinhas linhas e nColunas colunas, que deve ser retornada. A leitura deve ser feita de maneira a preencher uma linha da matriz de cada vez, isto é, se um vetor tem 5 linhas e 10 colunas, será lida a primeira linha, depois a segunda linha, e assim por diante.
- comparalmagem(imagem1,imagem2): devolve 1 se duas matrizes-imagens são iguais e 0, caso contrário (onde cada matriz-imagem é representada por uma array bidimensional).
- binarizalmagem(imagem, limiar): transforma os pixels em valores zero (se o valor atual for menor ou igual ao limiar) ou 1 (se o valor atual for menor que o limiar). Este método é útil quando desejamos obter uma matriz-imagem onde o fundo é composto por um único valor e o objeto a ser analisado é composto por outro valor. Por exemplo: fundo é composto por pixels com valor 0 e objeto com pixel com valor 1. É útil para eliminar ruídos em matrizes-imagens. Por exemplo: se for aplicado este método no exemplo da matriz-imagem anterior, com limiar = 4, o resultado será:



 calculaTamanhoBordaHorizontal(imagem, linha): devolve quantos pixels pertencem a uma borda horizontal localizada em uma determinada linha da matriz-



- imagem. Uma borda horizontal é uma sequência de dois ou mais algarismos diferentes de zeros presentes em uma linha da matriz-imagem.
- calculaTamanhoBordaVertical(imagem, coluna): devolve quantos pixels pertencem a uma borda vertical localizada em uma determinada coluna da matrizimagem. Uma borda vertical é uma sequência de dois ou mais algarismos diferentes de zeros presentes em uma coluna da matriz-imagem.

Classe Placa:

Esta classe contém métodos que auxiliam no reconhecimento de dígitos de uma placa de carro.

A classe Placa deve implementar pelo menos os seguintes métodos:

- inicializaPlaca(imagem1, imagem2, imagem3, imagem4): recebe 4 matrizes-imagens, uma referente a cada dígito da placa.
- analisaPlaca(): verifica 4 matrizes-imagens referentes aos 4 números presentes em placas de veículos e devolve um número de 4 algarismos.
- **verificaDigito (imagem)**: recebe uma matriz-imagem e devolve o dígito que ela significa.

Classe Veiculo:

Esta classe contém métodos que armazenam e recuperam a placa e o proprietário do veículo.

A classe Veiculo deve implementar pelo menos os seguintes métodos:

- armazena Veiculo (proprietario, placa, cidade, estado): armazena os dados do veículo
- devolveDados(): devolve os dados do veículo (proprietário, placa, cidade e estado).

Classe SistemaDeldentificacaoVeicular:

Esta classe deve usar as classes anteriores para instanciar objetos, criando veículos com seus dados e verificando suas placas. Deve ter um método *main* que armazena dados dos veículos, recebe placas de veículos como vetores-imagens e imprime os dados dos proprietários dos veículos identificados. Os dados dos veículos identificados são obtidos através do método:

 devolveDados(placa): a partir da placa de um veículo, devolve os dados armazenados.

Os dados dos veículos a serem cadastrados devem ser lidos de um arquivo de entrada – dados de um mesmo veículo em uma mesma linha. As placas recebidas como imagens para consulta devem também ser lidas de um outro arquivo de entrada, com 1 imagem (1 vetor-imagem) em cada linha (portanto a imagem completa de uma placa de carro a cada 4 linhas, sendo 4 vetores-imagens – um para cada dígito). Após transformação do vetor-imagem em matriz-imagem, essas devem ser processadas usando os métodos apresentados e o programa deve devolver os dados do veículo (proprietário, cidade e estado) a partir de uma placa fornecida. Se o veículo não estiver cadastrado, deve imprimir uma mensagem informando o fato. O nome dos dois arquivos



de entrada devem ser lidos da linha de comando na chamada do programa. Será fornecido um exemplo de cada arquivo para teste do seu programa.



Exemplos:

• A chamada do programa deve ser algo semelhante a:

java SistemaDeldentificacaoVeicular cadastroPlacas.txt consultaDePlacas.txt

onde:

cadastroPlacas.txt é o nome de um aquivo contendo várias linhas, cada uma da forma:

NomeDoProprietario placa cidade estado

Ex:

"Fatima L. S. Nunes" 1736 Bauru SP

"Ariane Machado Lima" 4258 "Taubaté" SP

"Willian Honda" 7645 "Rio de Janeiro" RJ

consultaDePlacas.txt é o nome de um arquivo contendo grupos de 5 linhas, cada uma da forma:

Carro numeroConsulta quantLin quantCol vetorImagemDigito1 quantLin quantCol vetorImagemDigito2 quantLin quantCol vetorImagemDigito3 quantLin quantCol vetorImagemDigito4

Ex:

Carro 1

10 20 74 20 58 02 57 29 87 52 70 10 97 57 29 37 57 57 92 17 37 58 72

30 40 76 97 98 76 65 64 56 46 76 87 08 67 56 45 35 47 58 67 87 09 09 09 09 43 54...

15 15 78 67 68 97 00 07 01 36 61 37 56 66 66 60 16 75 21 75 41 64 56 46 28 76 72 56 37...

15 10 62 78 56 72 75 65 64 80 26 78 68 76 86 86 57 81 96 76 75 45 34 23...

Carro 2

30 40 76 97 98 76 65 64 56 46 76 87 08 67 56 45 35 47 58 67 87 09 09 09 09 43 54...

15 20 74 20 58 02 57 29 87 52 70 10 97 57 29 37 57 57 92 17 37 58 72

20 25 78 67 68 97 00 07 01 36 61 37 56 66 66 60 16 75 21 75 41 64 56 46 28 76 72 56 37...

15 23 62 78 56 72 75 65 64 80 26 78 68 76 86 86 57 81 96 76 75 45 34 23....

e a saída deve ser algo da forma: RESULTADO DA CONSULTA:

Carro NumeroConsulta:

Proprietário: NomeDoProprietário

Placa: NúmeroDaPlaca

Cidade: *cidade* Estado: *estado*

Ex:

RESULTADO DA CONSULTA:

Carro 1:



Proprietário: Fátima L. S. Nunes

Placa: 1736 Cidade: Bauru Estado: SP

Carro 2:

Veículo não cadastrado!

OBSERVAÇÕES SOBRE A ENTREGA DOS TRABALHOS:

- Os critérios de correção utilizados serão: corretude (resultados corretos), clareza e documentação do código fonte.
- Deverão ser postados no sistema COL um zip com os arquivos .java e .class.
- Os arquivos devem ser compactados em um único arquivo com o nome nrUSP_nomeCompletoSemEspaçosMasComPrimeirasLetrasEmMaiusculoEstiloJ ava.zip. Arquivos fora deste formato terão desconto de nota!
- Somente o arquivo ZIP deve ser postado no sistema COL.
- A responsabilidade de postagem no sistema COL é exclusiva do aluno. Por isso, problemas referentes ao uso do sistema devem ser resolvidos *com antecedência*.
- Em hipótese alguma será postergada a data de postagem no sistema.

