PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

Avaliação 4 – Fundamentos de programação

Nome: *Giovane Bianchi Milani* ***Data:*** *23/06/2021*

1 Itens do menu

1.1 Visualizar mapa do estacionamento

```
public static void desenharMapa () {
    /**
     * Imprime na tela o mapa do estacionamento
     * ==========
     * @return void
     */
    String[] fileiras = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J"};

    System.out.println("\n========== ESTACIONAMENTO ==========");
    System.out.println("\t■ - Vaga ocupada\n");

    System.out.println("_____1___2___3___4___5___6___7___8___9___10___11_");

    // Passando por todas as linhas
    for (int linha=0; linha<mapa.length; linha++) {
```

Parte 1 – desenhaMapa()

Para cada fileira do estacionamento a função executará diversos passos, são eles:

a) Imprime a numeração da fileira, de acordo com o índice no array constante “fileiras”, e logo após uma linha divisória.

b) Para cada coluna nessa fileira imprime linhas divisórias para a vaga, que caso estiver livre, ficará com o meio vazio, caso contrário, estará preenchida com o símbolo “■” .

```
// Passando por todas as linhas
for (int linha=0; linha<mapa.length; linha++) {
    // Imprimindo numeração fileira
    System.out.print("  "+fileiras[linha]+" ");
    // Imprimindo vagas
    System.out.print("|");
    for (int coluna=0; coluna<mapa[linha].length; coluna++) {
        boolean vaga = mapa[linha][coluna];
        if (vaga) {
            System.out.print("   |"); // vaga livre
        }
        else {
            System.out.print(" ■ |"); // vaga ocupada
        }
    }
```

Parte 2 – desenharMapa()

c) Começa uma nova linha e imprime a divisória no espaço ocupado pela numeração da fileira.

d) Novamente, para cada coluna, imprime as divisórias da vaga, nesse caso, todas estarão vazias, pois são uma segunda camada para a vaga, já impressa na linha de cima.

```
// Nova linha
System.out.println();
// Imprime o espaço do código da fileira
System.out.print("____");

// Imprime segunda linha para a vaga
for (int i = 0; i<mapa[linha].length+1; i++) {
    System.out.print("|");
    if (i < mapa[linha].length) {
        System.out.print("___");
    }
}
```

Parte 3 – desenharMapa()

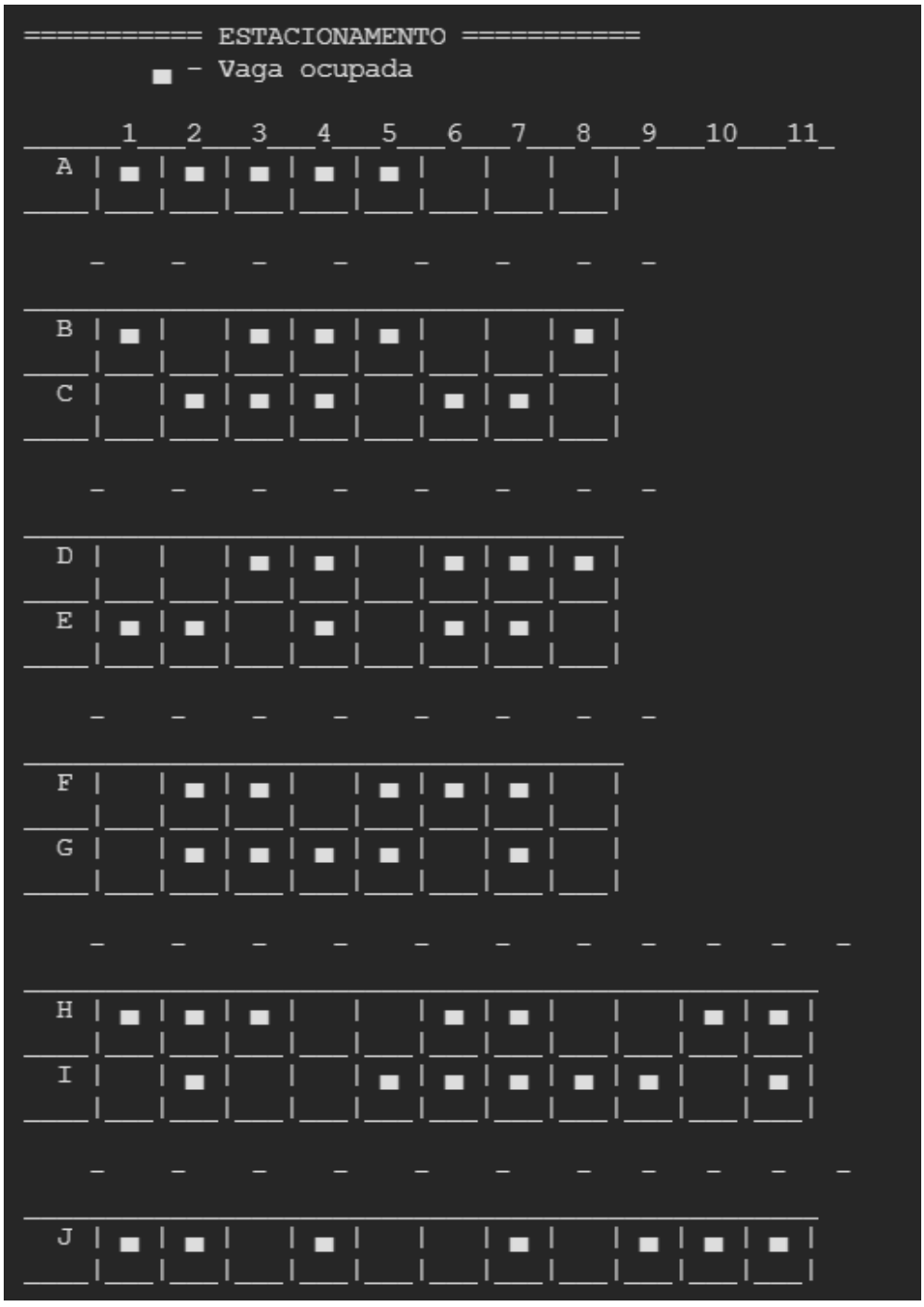
e) Caso o índice da linha seja par, outra linha será iniciada, e uma rua será impressa, que fica entre as fileiras.

```
// Imprime rua
if (linha % 2 == 0) {
    System.out.println("\n");
    if (linha < 6) {
        System.out.println("   -   -   -   -   -   -   -   -");
        System.out.println("_____________________________________");
    }
    else {
        System.out.println("   -   -   -   -   -   -   -   -   -   -   -");
        System.out.println("_____________________________________________");
    }
}
else {
    System.out.println();
}
```

Parte 4 – desenharMapa()

Resumindo, a função executa todos esses passos demonstrados acima, para cada fileira do estacionamento.

Com isso, teremos esse resultado no console:



Resultado – desenharMapa()

Obs.: O mapa acima não corresponde com os resultados dos outros métodos que serão abordados e demonstrados neste relatório. Isso acontece, pois, as vagas ocupadas são geradas aleatoriamente a cada iniciação do código.

1.2 Ocupar uma vaga

Recebe com parâmetro o código da vaga no formato letra e número, executando os seguintes passos:

a) Converte a string em um array, usando o método “split()”.

b) Identifica a linha, usando o método “indexOf()” na string das fileiras, passando como parâmetro o primeiro item do array do código, ou seja, irá retornar o índice em que o caractere está na string das fileiras, que é correspondente no estacionamento.

c) Identifica a coluna, usando o operador ternário, caso o comprimento do array código seja 2, a coluna será o segundo item do array diminuído de 1 unidade, caso o comprimento seja 3, ou seja, coluna de 2 dígitos, irá concatenar o segundo e o terceiro dígito do array e atribuirá à coluna diminuído de 1 unidade. Obs.: Foi usado o método “parseInt()” da classe “Integer”, para converter string para número inteiro.

d) Com a linha e coluna identificadas, essa vaga é verificada, caso esteja livre, será definida como ocupada, caso contrário, irá retornar uma mensagem ao usuário.

```java
public static void ocuparVaga (String codigo_vaga) {
    /**
     * Ocupa a vaga especificada caso esteja disponível,
     * caso contrário, exibi uma mensagem para o usuário
     * ==================
     * @param codigo_vaga: código da vaga que será ocupada
     * @return void
     */
    String fileiras = "ABCDEFGHIJ";

    // Lendo o código
    String[] codigo = codigo_vaga.split(""); // Divide todos caracteres do código em um array
    int linha = fileiras.indexOf(codigo[0].toUpperCase()); // Índice na string fileiras é correspondente a linha do mapa
    // Identifica se a coluna é de um ou dois dígitos e converte para int da forma correta
    int coluna = (codigo.length == 2) ? Integer.parseInt(codigo[1])-1 : Integer.parseInt(codigo[1] + codigo[2])-1;

    // Verifica se a vaga está livre, nessa caso, editaria ela para ocupada
    if (mapa[linha][coluna]) {
        System.out.printf("\nVaga %s foi ocupada com sucesso!\n", codigo_vaga);
        mapa[linha][coluna] = false;
    }
    // Se não estiver, imprime uma mensagem para o usuário
    else {
        System.out.printf("\nA vaga %s já está ocupada...\n", codigo_vaga);
    }
}
```

Código completo - ocuparVaga()

Resultados ao informar uma vaga livre e ao informar uma vaga já ocupada, respectivamente:

```
Digite o código da vaga que deseja ocupar:
> J11

Vaga J11 foi ocupada com sucesso!
```

Resultado 1 – ocuparVaga()

```
Digite o código da vaga que deseja ocupar:
> A1

A vaga A1 já está ocupada...
```

Resultado 1 – ocuparVaga()

1.3 Liberar uma vaga

Recebe como parâmetro o código da vaga, no formato letra e número, e para identificar a posição da vaga, usa a mesma lógica do método “ocuparVaga()”. No fim, verifica se a vaga já está livre ou não, imprimindo uma mensagem para o usuário conforme o resultado do teste.

```java
public static void liberarVaga (String codigo_vaga) {
    /**
     * Libera a vaga especificada
     * ==================
     * @param codigo_vaga: código da vaga que será ocupada
     * @return void
     */
    String fileiras = "ABCDEFGHIJ";

    // Lendo o código
    String[] test = codigo_vaga.split(""); // Divide todos caracteres do código em um array
    int linha = fileiras.indexOf(test[0].toUpperCase()); // Índice na string fileiras é correspondente a linha do mapa
    // Identifica se a coluna é de um ou dois dígitos e converte para int da forma correta
    int coluna = (test.length == 2) ? Integer.parseInt(test[1])-1 : Integer.parseInt(test[1] + test[2])-1;

    // Verifica se a vaga está livre, nesse caso retorna uma mensagem para o usuário
    if (mapa[linha][coluna]) {
        System.out.printf("\nA vaga %s já está livre...\n", codigo_vaga);
    }
    // Se estiver ocupada, torna a vaga livre
    else {
        System.out.printf("\nVaga %s foi liberada com sucesso!\n", codigo_vaga);
        mapa[linha][coluna] = true;
    }
}
```

Código completo – liberarVaga()

Ao passar uma vaga ocupada e uma livre, temos respectivamente como resultado:

```
Digite o código da vaga que deseja liberar:
> I9

Vaga I9 foi liberada com sucesso!
```

Resultado 1 – liberarVaga()

```
Digite o código da vaga que deseja liberar:
> A5

A vaga A5 já está livre...
```

Resultado 2 – liberarVaga()

1.4 Encontrar primeira vaga livre

Para cada linha e coluna da matriz do estacionamento, verifica se a posição está livre e se nenhuma outra posição já foi encontrada, caso atenda as duas proposições, irá marcar a vaga como ocupada e imprimir uma mensagem para o usuário com o código da vaga encontrada.

Obs.: No início do método é atribuída uma variável booleana como false, e quando a primeira vaga é encontrada ela é definida como true, com isso, o método só irá ocupar a primeira vaga livre que encontrar, mesmo que continue passando pelo restante das vagas.

No fim, caso atinja a última linha e última coluna e nenhuma vaga tiver sido encontrada, irá imprimir uma mensagem para o usuário.

```java
public static void vagaLivre () {
    /**
     * Procura pela primeira vaga livre, marca-a como ocupada
     * e imprime ela na tela
     * ==================
     * @return void
     */
    String[] fileiras = {"A","B","C","D","E","F","G","H","I","J"};
    String codigo;
    boolean vagaEncontrada = false;

    for (int linha = 0; linha < mapa.length; linha++) {
        for (int coluna = 0; coluna < mapa[linha].length; coluna++) {
            // Se a vaga estiver livre e nenhuma vaga tiver sido encontrada, torna a vaga ocupada
            if (mapa[linha][coluna] && !vagaEncontrada) {
                mapa[linha][coluna] = false;
                vagaEncontrada = true;
                codigo = ""+fileiras[linha]+(coluna+1);
                System.out.printf("\nVaga encontrada... %s foi marcada como ocupada...\n", codigo);
            }
        }
        /**Se tiver passado por todas as vagas e nenhuma tiver sido encontrada
         retorna uma mensagem pro usuário*/
        if (linha == mapa.length-1 && !vagaEncontrada) {
            System.out.printf("\nNenhuma vaga encontrada... Estaciomento lotado...\n");
        }
    }
}
```

Código completo - vagaLivre()

Como resultado no console temos:

```
Vaga encontrada... A2 foi marcada como ocupada...
```

```
Nenhuma vaga encontrada... Estaciomento lotado...
```

1.5 Encontrar vaga especial

Utiliza da mesma lógica do método “vagaLivre()”, porém começa a verificação na segunda coluna de cada linha e ainda testa se a vaga anterior também está livre, desse modo, irá definir como ocupada as primeiras duas vagas livres seguidas que achar, imprimindo-as com uma mensagem para o usuário.

```java
public static void vagaEspecial () {
    /**
     * Procura por vagas especiais, ou seja, adjacentes,
     * e marca elas como ocupada, imprimindo-as na tela
     * ==================
     * @return void
     */
    String[] fileiras = {"A","B","C","D","E","F","G","H","I","J"};
    String codigo;
    String codigo2;
    boolean vagaEncontrada = false;

    for (int linha = 0; linha < mapa.length; linha++) {
        // Começando na segunda coluna
        for (int coluna = 1; coluna < mapa[linha].length; coluna++) {
            // Se a vaga estiver livre e nenhuma ainda não tiver sido encontrada
            if (mapa[linha][coluna] && !vagaEncontrada) {
                // verifica se a vaga anterior também está livre
                if (mapa[linha][coluna-1]) {
                    // Definindo as vagas como ocupadas
                    mapa[linha][coluna] = false;
                    mapa[linha][coluna-1] = false;
                    vagaEncontrada = true;
                    // Definindo os códigos das vagas
                    codigo = ""+fileiras[linha]+(coluna+1);
                    codigo2 = ""+fileiras[linha]+(coluna);
                    // Mensagem para o usuário
                    System.out.printf("\nVaga especial encontrada...\n%s e %s foram marcadas como ocupadas...\n", codigo2, codigo);
                }
            }
        }
        /**Se tiver passado por todas as vagas e nenhuma tiver sido encontrada
         retorna uma mensagem pro usuário*/
        if (linha == mapa.length-1 && !vagaEncontrada) {
            System.out.printf("\nNenhuma vaga especial encontrada...\n");
        }
    }
}
```

Código completo - vagaEspecial()

Como resultado temos:

```
Vaga especial encontrada...
D2 e D3 foram marcadas como ocupadas...
```

```
Nenhuma vaga especial encontrada...
```

1.6 Exibir estatísticas

Inicialmente, declara uma variável do tipo inteira para o total de vagas e atribuí duas outras com zero, uma para a contagem de vagas livres e a outra para ocupadas. Com isso, passa por todas linhas e colunas da matriz, verificando se a vaga está livre ou ocupada e acrescenta uma unidade conforme o resultado da proposição.

No fim, imprime os resultados para o usuário.

```java
public static void estatisticas () {
    /**
     * Imprime na tela o percentual de vagas ocupadas e
     * de vagas livres
     * ==================
     * @return void
     */
    int total_vagas;
    int vagas_livres=0;
    int vagas_ocupadas=0;

    // Para cada vaga
    for (int linha = 0; linha < mapa.length; linha++) {
        for (int coluna = 0; coluna < mapa[linha].length; coluna++) {
            // Verifica se está livre ou não
            if (mapa[linha][coluna]) {
                vagas_livres++;
            }
            else {
                vagas_ocupadas++;
            }
        }
    }
    total_vagas = vagas_livres + vagas_ocupadas;

    //Imprime as estastísticas na tela
    System.out.println("\t\t=== ESTATÍSTICAS ===");
    System.out.printf("Vagas livres: %d\tPercentual de vagas livres: %.2f%%", vagas_livres, 100.0*vagas_livres/total_vagas);
    System.out.printf("\nVagas ocupadas: %d\tPercentual de vagas ocupadas: %.2f%%\n", vagas_ocupadas, 100.0*vagas_ocupadas/total_vagas);
}
```

Código completo - estatisticas()

Como resultado:

```
                === ESTATÍSTICAS ===
Vagas livres: 33        Percentual de vagas livres: 37,08%
Vagas ocupadas: 56      Percentual de vagas ocupadas: 62,92%
```

Resultado – estatisticas()

2. Funções extras

Função “main” e funções auxiliares para os demais métodos do programa.

2.1 Imprimir menu principal

Por questões de organização, foi criado um método separado para imprimir os itens do menu principal.

```java
public static void imprimirMenuPrincipal () {
    /**
     * Imprime o menu principal
     * ==================
     * @return void
     */
    System.out.println("\n===== GERENCIADOR DE ESTACIONAMETO =====");
    System.out.println("\t === Menu principal ===\n");
    System.out.println("Escolha oque deseja fazer:");
    System.out.println("1. Visualizar mapa do estacionamento");
    System.out.println("2. Ocupar uma vaga");
    System.out.println("3. Liberar uma vaga");
    System.out.println("4. Encontrar a primeira vaga livre");
    System.out.println("5. Encontrar vaga especial");
    System.out.println("6. Exibir estatísticas");
    System.out.println("7. Sair do programa");
    System.out.println("Digite a sua escolha:");

}
```

Código completo - imiprimirMenuPrincipal()

```
===== GERENCIADOR DE ESTACIONAMETO =====
         === Menu principal ===

Escolha oque deseja fazer:
1. Visualizar mapa do estacionamento
2. Ocupar uma vaga
3. Liberar uma vaga
4. Encontrar a primeira vaga livre
5. Encontrar vaga especial
6. Exibir estatísticas
7. Sair do programa
Digite a sua escolha:

>
```

Resultado – imprimirMenuPrincipal()

2.2 Criar mapa de estacionamento

Função que gera a matriz do mapa do estacionamento, com os seguintes passos:

a) Primeiramente inicia uma matriz com apenas as linhas definidas, no caso são dez.

b) Para cada linha da matriz, defini a quantidade de colunas, oito até a linha sete e onze para as três últimas, com isso, defini o valor de todas as colunas dessa linha para true, ou seja, vaga está livre.

c) Após isso, ainda no mesmo “for” das fileiras, enquanto 70% das vagas da linha não estão ocupadas, ficará gerando índices de coluna aleatórios e caso esteja livre, definirá a vaga como false, ou seja, ocupada.

```java
public static boolean[][] criarMapa () {
    /**...
    boolean[][] map = new boolean[10][];
    // Atribui para cada fileira do mapa a qtd de vagas
    int qtd_vagas;
    for (int linha=0; linha<map.length; linha++) {
        // Se a linha for menor que 7, a qtd de vagas será 8, senão será 11
        qtd_vagas  = (linha < 7) ? 8 : 11;
        map[linha] = new boolean[qtd_vagas];

        // 'Preenche' todas as posições com true, ou seja, vaga livre
        for (int coluna=0; coluna<map[linha].length; coluna++) {
            map[linha][coluna] = true;
        }

        // Gera números aleatórios para posições até que cerca
        // de 70% das vagas na linha estejam ocupadas
        int colunaAleatoria = 0;
        int cont = 0;
        while (cont < map[linha].length * 70 / 100) {
            colunaAleatoria = (int)(Math.random() * map[linha].length);
            if (map[linha][colunaAleatoria]) {
                map[linha][colunaAleatoria] = false;
                cont++;
            }
        }
    }
    return map;
}
```

Código completo - criarMapa()

Desse modo, o método retorna uma matriz, mais especificamente um “Jagged array”, booleana, em que true representa vaga livre e false vaga ocupada.

2.3 Sair do sistema

Método que é chamado para quebrar o loop de funcionamento do programa, define o atributo de classe “rodando” para false e imprime uma mensagem para o usuário.

```java
public static void sair () {
    /**
     * Altera o atributo de classe 'rodando' para false
     * encerrando assim o loop do programa
     * ==========
     * @return void
     */
    rodando = false;
    System.out.println("Saindo do sistema...");
}
```

Código completo - sair()

2.4 Main

Contém o loop de funcionamento do programa, que enquanto o atributo de classe “rodando” for verdadeiro, continuará pedindo as ações para o usuário e rodando o código, assim que o método “sair()” for chamado, o atributo “rodando” será definido como falso e o loop se encerrará, saindo do programa.

```java
Run | Debug
public static void main (String[] args) {
    /**
     * Contém o loop de funcionamento do programa
     */

    Scanner in = new Scanner(System.in);
    int opcao;
    String codigo;

    // Loop de funcionamento
    while (rodando) {

        imprimirMenuPrincipal();

        // Perguntando ao usuário
        do {
            System.out.print("\n> ");
            opcao = in.nextInt();
        } while (opcao < 1 || opcao > 7);
```

Parte 1 – main()

```
        // Chamando a função solicitada
        switch (opcao) {
            case 1:
                desenharMapa();
                break;
            case 2:
                System.out.println("Digite o código da vaga que deseja ocupar: ");
                System.out.print("> ");
                codigo = in.next();
                ocuparVaga(codigo);
                break;
            case 3:
                System.out.println("Digite o código da vaga que deseja liberar: ");
                System.out.print("> ");
                codigo = in.next();
                liberarVaga(codigo);
                break;
            case 4:
                vagaLivre();
                break;
            case 5:
                vagaEspecial();
                break;
            case 6:
                estatisticas();
                break;
            case 7:
                in.close();
                sair();
                break;
        } // switch case
    } // loop
}
```

Parte 2 – main()