UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Estruturas de Dados I

Laboratório – Aplicação de Deque como Fila – Demultiplexação

Atividade em Grupo (duas a três pessoas) [1]

Nome do Integrante (ordem alfabética)	TIA
Erik Samuel Viana Hsu	32265921
Mateus Kenzo Iochimoto	32216289
Rodrigo Machado De Assis Oliveira De Lima	32234678
Thiago Shihan Cardoso Toma	32210744

Conteúdo

O relatório a seguir **deve conter dois testes de cada item do menu**. Além disso, ao final dele, colocar um Apêndice com os códigos fontes criados.

Ao enviar pelo Moodle, não se esquecer de incluir o relatório e os códigos fontes criados.

Relatório

Testes do item 1 do menu:

Teste de um caso de sucesso do programa

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:1
Lê canal compartilhado:
Digite o número do identificador: 1
Digite o valor: 12
Digite o número do identificador: 2
Digite o valor: 36
Digite o número do identificador: 3
Digite o valor: 89
Digite o número do identificador: -1
Leituras e inserções finalizadas

Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:
```

[1] Atividade adaptada do material do prof. Dr. Jean M. Laine.

Teste de um caso onde o usuário coloca um número de identificador inválido.

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:1
Lê canal compartilhado:
Digite o número do identificador: 2
Digite o valor: 78
Digite o número do identificador: 1
Digite o valor: 23
Digite o número do identificador: 1
Digite o valor: -1
Digite o número do identificador: 3
Digite o valor: 42
Digite o número do identificador: 9
Erro: Os números dos identificadores vão apenas de 1 a 3. A lista será zerada.

Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:
```

Testes do item 2 do menu:

Teste normal da opção

```
Lê canal compartilhado:
Digite o número do identificador: 3
Digite o valor: 456
Digite o número do identificador: 2
Digite o valor: 54
Digite o número do identificador: 1
Digite o valor: 85
Digite o número do identificador: 2
Digite o valor: 62
Digite o número do identificador: 3
Digite o valor: 64
Digite o número do identificador: 1
Digite o valor: 64
Digite o número do identificador: 3
Digite o valor: 12
Digite o número do identificador: 2
Digite o valor: 787
Digite o número do identificador: -1
Leituras e inserções finalizadas

Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:2
Imprime canal compartilhado:
[[3,456],[2,54],[1,85],[2,62],[3,64],[1,64],[3,12],[2,787],
Menu de Opções
```

Teste demonstrando que a saída continua a mesma após repetir a opção múltiplas vezes

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:2
Imprime canal compartilhado:
[[3,456],[2,54],[1,85],[2,62],[3,64],[1,64],[3,12],[2,787],
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:2
Imprime canal compartilhado:
[[3,456],[2,54],[1,85],[2,62],[3,64],[1,64],[3,12],[2,787],
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra
```

Testes do item 3 do menu:

Teste de caso 3 bem sucedido:

```
Imprime canal compartilhado:
[[1,6],[2,7],[3,8]]
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:3
Desefileira canal compartilhado:
Canal desenfileirado
```

Teste de caso 3 em caso de canal compartilhado vazio:

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:3
Desefileira canal compartilhado:
Canal compartilhado vazio
```

Testes do item 4 do menu:

Teste normal da opção:

```
Main (4) [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.just

Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:4
Imprime as filas geradas:
Fluxo 1:[10|22|37|14]
Fluxo 2:[98|32|42]
Fluxo 3:[17|66|22|98]
```

Teste caso os 3 fluxos estiverem vazios:

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:4
Imprime as filas geradas:
Fluxos vazios
```

Teste com um dos fluxos vazios:

```
Main (4) [Java Application] C:\Users\user\.p2\pool\plugi
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:4
Imprime as filas geradas:
Fluxo 1 vazio
Fluxo 2:[5]
Fluxo 3:[6]
```

Testes do item 5 do menu:

```
Menu de Opções

1 - Lê canal compartilhado
2 - Imprime canal compartilhado
3 - Desefileira canal compartilhado
4 - Imprime as filas geradas
5 - Encerra

 Opção:5
Programa encerrado
```

Códigos fontes criados

Main.java:

```
/
*
    NOME: ERIK SAMUEL VIANA HSU - TIA: 32265921
    NOME: MATEUS KENZO IOCHIMOTO - TIA: 32216289
    NOME: RODRIGO MACHADO DE ASSIS OLIVEIRA DE LIMA - TIA: 32234678
    NOME: THIAGO SHIHAN CARDOSO TOMA - TIA: 32210744
    */


    import java.util.InputMismatchException;
    import java.util.Scanner;


    class Main {
     public static void main(String[] args) {
       Deque d = new Deque();
       Deque f1 = new Deque();
       Deque f2 = new Deque();
       Deque f3 = new Deque();
       String opcoes = "\nMenu de Opções\n\n1 - Lê canal compartilhado\n2 - Imprime canal compartil
- Desefileira canal compartilhado\n4 - Imprime as filas geradas\n5 - Encerra\n\n Opção:";
       Scanner ent = new Scanner(System.in);
       int opcao = 0;
       do{
         System.out.print(opcoes);
         opcao = ent.nextInt();
         switch(opcao){
           case 1:
             System.out.println("Lê canal compartilhado: ");
             //Parte inicial da opção 1
```

```
        Scanner s = new Scanner(System.in);

        //Receber o valor inicial dos identificadores:

        System.out.print("Digite o número do identificador: ");

        int id = 0;

        //Tratamento de erro caso a entrada digitada seja de algum tipo primitivo sem ser inte

        try {

                    id = s.nextInt();

              }

        catch(InputMismatchException e) {

              System.out.println("Erro: Digite um valor do tipo inteiro!");

              break;

        }

        //Caso o identificador seja -1, finaliza as leituras e inserções na fila

        if (id == -1) {

              System.out.println("Leituras e inserções finalizadas");

              break;

        }

        //Caso o identificador tenha valor < que 1 ou maior que 3, visto que trabalhamos com a
fluxos numerados de 1 a 3
        else if (id < 1 || id > 3) {

              System.out.println("Erro: Os números dos identificadores vão apenas de 1 a 3. A
será zerada.");
              while (d.isEmpty() == false) {

                    d.dequeueLeft();

                    d.dequeueRight();

              }

              break;

        }

        //Caso não haja erros, o número do identificador é enfileirado

        else

              d.enqueueLeft(id);

        //Receber o valor inicial dos valores:

        System.out.print("Digite o valor: ");
```

```
        int valor = 0;
        //Tratamento de erro caso a entrada do valor digitado seja de algum tipo primitivo sem
inteiro
        try{
              valor = s.nextInt();
        }
        catch(InputMismatchException e){
              System.out.println("Erro: Digite um valor do tipo inteiro!");
              break;
        }
        //Caso não haja erros, o valor digitado é enfileirado
        d.enqueueLeft(valor);
        //Loop de leitura e inserção na fila até que o usuário digite -1
        while (id > 0 && id <= 3 && id != -1) {
              System.out.print("Digite o número do identificador: ");
              //Tratamento de erro caso a entrada do valor digitado seja de algum tipo primiti
ser inteiro
            try{
              id = s.nextInt();
            }
            catch(InputMismatchException e){
              System.out.println("Erro: Digite um valor do tipo inteiro!");
              break;
            }
            //Caso o identificador seja -1, finaliza as leituras e inserções na fila
              if (id == -1) {
                    System.out.println("Leituras e inserções finalizadas");
                    break;
              }
              //Tratamento de erro caso o identificador digitado seja menor que 1 ou maior que
              else if (id < 1 || id > 3) {
                    System.out.println("Erro: Os números dos identificadores vão apenas de 1
lista será zerada.");
                    while (d.isEmpty() == false) {
```

```
                        d.dequeueLeft();
                        d.dequeueRight();
                    }
                    break;
                }
                //Caso não haja erros, o número do identificador é enfileirado
                d.enqueueLeft(id);
                System.out.print("Digite o valor: ");
                //Tratamento de erro caso a entrada do valor digitado seja de algum tipo primiti
ser inteiro
                try{
                    valor = s.nextInt();
                }
                catch(InputMismatchException e) {
                    System.out.println("Erro: Digite um valor do tipo inteiro!");
                break;
                }
              //Caso não haja erros, o valor digitado é enfileirado
                d.enqueueLeft(valor);
          }
          break;

        case 2:
          if (d.isEmpty() == true) {
                System.out.println("Erro: A fila correspondente está vazia!");
                break;
          }
          else
                System.out.println("Imprime canal compartilhado: ");
                int tam = d.size(); // variável guarda o size() para usar o valor no for
              String canal = "[";
              Deque temp = new Deque(); //deque temporario para guardar os valores do dequeue do
principal
```

```
            for (int i = 0; i < tam; i++){
              temp.enqueueRight(d.dequeueRight()); //guarda o valor
              if (i%2 == 0)
                    canal += "["+temp.getRight()+","; //guarda na string o valor guardado no
temporario
              if (i%2 != 0)
                if (i == tam-1){
                  canal += temp.getRight()+"]";
                }
                else{
                    canal += temp.getRight()+"],";
                }
            }
            canal += "]";
            System.out.print(canal); //printa a string com os valores do deque
            for (int i = 0; i < tam; i++){
              d.enqueueRight(temp.dequeueRight()); //guarda os valores de volta do deque princ
            }
            break;
      case 3:
        System.out.println("Desefileira canal compartilhado: ");
        if (d.isEmpty() == true) {
              System.out.println("Canal compartilhado vazio ");
              break;
        }
        else {
        int j = 0;
        int tamD = d.size(); // variável guarda o size() para usar o valor no while

        while ( j < tamD) {
          //Enfileira o valor na fila correta indicada pelo indicador
              if (d.getRight() == 1) {
                    d.dequeueRight();
```

```
                f1.enqueueLeft(d.dequeueRight());

        }

        else if (d.getRight() == 2) {

                d.dequeueRight();

                        f2.enqueueLeft(d.dequeueRight());

                }


        else if (d.getRight() == 3) {

                d.dequeueRight();

                f3.enqueueLeft(d.dequeueRight());

        }

    //incrementa indice de 2 em 2 para verificar os identificadores

        j+=2;

  }

  System.out.println("Canal desenfileirado ");

  break;

  }

case 4:

  System.out.println("Imprime as filas geradas: ");

  //Avisa caso todos os fluxos estiverem vazios

  if (f1.isEmpty() && f2.isEmpty() && f3.isEmpty()) {

        System.out.println("Fluxos vazios");

        break;

  }

  else {

        if(f1.isEmpty()) {

                System.out.println("Fluxo 1 vazio");

        }

        else {

                System.out.print("Fluxo 1:");

                System.out.print("[");

      //Enqunato fila nao esta vazia imprime cada elemento e desenfileira
```

```
		while(f1.isEmpty() == false) {
			if (f1.size()!= 1) {
				System.out.print(f1.dequeueRight()+"|");
			}
			else {
				System.out.println(f1.dequeueRight()+"]");
			}
		}


	}
	 if(f2.isEmpty()) {
		System.out.println("Fluxo 2 vazio");
	 }
	 else {
		System.out.print("Fluxo 2:");
		System.out.print("[");
	//Enqunato fila nao esta vazia imprime cada elemento e desenfileira
		while(f2.isEmpty() == false) {
			if (f2.size()!= 1) {
				System.out.print(f2.dequeueRight()+"|");
			}
			else {
				System.out.println(f2.dequeueRight()+"]");
			}
		}


	}
	 if(f3.isEmpty()) {
		System.out.println("Fluxo 3 vazio");
	 }
	 else {
```

```
                    System.out.print("Fluxo 3:");

                    System.out.print("[");

                //Enqunato fila nao esta vazia imprime cada elemento e desenfileira

                    while(f3.isEmpty() == false) {

                        if (f3.size()!= 1) {

                            System.out.print(f3.dequeueRight()+"|");

                        }

                        else {

                            System.out.println(f3.dequeueRight()+"]");

                        }

                    }



                }

            }


            break;

          case 5:

            System.out.println("Programa encerrado");

            ent.close();

            break;

        }

      }while (opcao != 5);


    }


  } //fim da Main
```

Deque.java:

```
public class
Deque {
                        private static final int TAM_DEQUE = 100;
```

```
private      int inicio, fim, qtde;

private int e[ ];

// Construtor que Inicia o Deque d

// como vazio e tamanho padrão

public Deque() {

       this(TAM_DEQUE);

}

// Construtor que Inicia o Deque d

// como vazio e tamanho informado

// pelo usuário da classe

public Deque(int tamanho) {

       this.inicio = 0;

       this.fim = 0;

       this.qtde = 0;

       this.e = new int [tamanho];

}

// Verifica se o Deque d está vazio

// retornando true (se vazio)

// e false (caso contrário)

public boolean isEmpty( ) {

       return qtde == 0;

}

// Verifica se o Deque d está cheio

// retornando true (se cheio)

// e false (caso contrário)



public boolean isFull( ) {

       return qtde == TAM_DEQUE;

}

// Obtém o elemento do início do Deque

public int getLeft ( ) {

       if (! isEmpty( )){
```

```
                return e[inicio];
            } else {
                System.out.println("deque empty");
                return -1;
            }
    }
    // Obtém o elemento do fim do deque D
    public int getRight ( ) {
            if (! isEmpty( )){
                if (fim == 0)
                  return e[TAM_DEQUE-1];
                else
                  return e[fim-1];
            } else {
                   System.out.println("deque empty");
                return -1;
            }
    }
    // Insere no "início-1" do Deque D
    public void enqueueLeft ( int e ) {
            if (! isFull( )){
                if (inicio == 0){
                  this.e[TAM_DEQUE-1] = e;
                  inicio = TAM_DEQUE-1;
                } else this.e[--inicio] = e;
                qtde++;
              } else
                     System.out.println("deque overflow");

    }
    // Insere no "fim" do Deque D
    public void enqueueRight ( int e ) {
            if (! isFull( )){
```

```
                this.e[fim++] = e;
                fim = fim % TAM_DEQUE;
                qtde++;
            } else
                System.out.println("deque overflow");

    }
    // Remove e retorna um elemento do início do Deque d
    public int dequeueLeft( ) {
            int aux;
            if (! isEmpty( )){
               aux = e[inicio];
               inicio = ++inicio % TAM_DEQUE;
               qtde--;
               return aux;
            }else{
                   System.out.println("deque underflow");
                return -1;
            }
    }
    // Remove e retorna um elemento  do final do Deque d
    public int dequeueRight( ) {
             int aux;
             if (! isEmpty( )){
               if (fim == 0) {
                 aux = e[TAM_DEQUE-1];
                 fim = TAM_DEQUE-1;
               } else {
                 aux = e[fim-1];
                 fim--;
               }
               qtde--;
```

```
                return aux;
            }else{
                System.out.println("deque underflow");
                return -1; }
    }
    // Retorna o total de elementos
    // armazenados no deque
    public int size() {
        return qtde;
    }

    @Override
    public String toString() {

        int indiceNovo = (inicio + qtde) % e.length;

        StringBuilder sb = new StringBuilder();
        sb.append("[Deque] quantidade: ")
            .append(qtde)
            .append(", capacidade: ")
            .append(e.length);
        if (qtde != 0) {
            sb.append(", primeiro (Esquerda): ")
                .append(getLeft())
                .append(", último (Direita): ")
                .append(getRight());
        }

        sb.append("\nConteudo do Deque': [ ");
        if (qtde != 0) {
            if (indiceNovo <= inicio) {
                for (int i = inicio; i < e.length; ++i)
```

```
                                        sb.append("[" + e[i] + "]");

                                for (int i = 0; i < indiceNovo; ++i)

                                        sb.append("[" + e[i] + "]");

                        } else {

                                for (int i = inicio; i < indiceNovo;
++i)

                                        sb.append("[" + e[i] + "]");

                        }

                }

                sb.append(" ]");

                return sb.toString();

        }

}
```