TT11 – Unidade VI:

Tipos Abstratos de Dados Flexíveis - Fila:

David de Sá Vieira de Faria – 699415

**Obs: todos códigos disponíveis no diretório ./Exercicios\_praticos/Unidade06c**

**Exercício – Slide 26:**

Seja nossa Fila, implemente um método para mostrar os elementos da fila.

|  |
| --- |
| public void mostrar(){      System.out.print("[ ");      for(Celula i = primeiro.prox; i != null; i = i.prox){          System.out.print(" " + i.elemento);      } // end for      System.out.print(" ]\n");  } // end mostrar () |

**Exercício – Slide 27:**

Seja nossa Fila, faça um método que retorne o maior elemento contido na mesma.

|  |
| --- |
| public int getMaior() throws Exception{      int elemento = -1;      if(primeiro == ultimo){          throw new Exception("Erro: primeiro igual ao ultimo");      } else{          elemento = primeiro.prox.elemento;          for(Celula i = primeiro.prox.prox; i != null; i = i.prox){              if(elemento < i.elemento)                  elemento = i.elemento;          } // end for      } // end else      return elemento;  } // end getMaior() |

**Exercício – Slide 29:**

Seja nossa Fila, faça um método para mostrar o terceiro elemento supondo que o mesmo existe.

|  |
| --- |
| public int getThirdElement(){      return(primeiro.prox.prox.prox.elemento);  } // end getThirdElement() |

**Exercício – Slide 31:**

Seja nossa Fila, faça um método que soma o conteúdo dos elementos contidos na mesma.

|  |
| --- |
| public int getSum(){      int soma = 0;      for(Celula i = primeiro.prox; i != null; i = i.prox){          soma += i.elemento;      } // end for      return soma;  } // end getSum() |

**Exercício – Slide 33:**

Seja nossa Fila, faça um método que inverta a ordem dos seus elementos.

|  |
| --- |
| void inverter () {      Celula fim = ultimo;      while (primeiro != fim){          Celula nova = new Celula (primeiro.prox.elemento);          nova.prox = fim.prox;          fim.prox = nova;          Celula tmp = primeiro.prox;          primeiro.prox = tmp.prox;          nova = tmp = tmp.prox = null;          if (ultimo == fim)              ultimo = ultimo.prox;      } // end while      fim = null;  } // end inverter() |

**Exercício – Slide 34:**

Seja nossa Fila, faça um método recursivo para contar o número de elementos pares AND múltiplos de cinco contidos na fila.

|  |
| --- |
| public int countElements(Celula i){      int count = 0;      if(i != null){          if((i.elemento % 2 == 0) || (i.elemento % 5 == 0)){              count += countElements(i.prox);              count++;          } else{              count += countElements(i.prox);          } // end else      } // end if      return count;  } // end countElements() |

**Exercício – Slide 38:**

Implemente o método Celula toFila(Celula topo) que recebe o endereço de memória da primeira posição de uma pilha flexível e retorna o endereço de memória do nó cabeça de uma fila flexível contendo os elementos da pilha na ordem em que os mesmos foram inseridos na pilha. Seu método deve percorrer a pilha e inserir cada elemento da mesma na nova fila a ser retornada. A pilha não pode ser destruída.

|  |
| --- |
| public Fila fromPilhaInvertida\_to\_Fila(Celula topo){      int n = 0, j = 0;      Fila newFila = new Fila();        for(Celula i = topo; i != null; i = i.prox)          n++;      int []tmp = new int[n];        for(Celula i = topo; i != null; i = i.prox, j++)          tmp[j] = i.elemento;        for(int i = n-1; i >= 0; i--)          newFila.inserir(tmp[i]);      tmp = null;      return newFila;  } // end fromPilhaInvertida\_to\_Fila() |

**Exercício – Slide 39:**

Implemente a fila flexível sem nó cabeça.

|  |
| --- |
| public class FilaSemCabeca {      private Celula primeiro;      private Celula ultimo;      /\*\*       \* Construtor da classe que cria uma fila sem elementos (sem no cabeca).       \*/      public FilaSemCabeca() {          primeiro = new Celula();          ultimo = primeiro;      } // end FilaSemCabeca()      /\*\*       \* Insere elemento na fila (politica FIFO).       \* @param x int elemento a inserir.       \*/      public void inserir(int x) {          ultimo.elemento = x;          ultimo.prox = new Celula();          ultimo = ultimo.prox;      } // end inserir()      /\*\*       \* Remove elemento da fila (politica FIFO).       \* @return Elemento removido.       \* @trhows Exception Se a fila nao tiver elementos.       \*/      public int remover() throws Exception {          if (primeiro == ultimo) {              throw new Exception("Erro ao remover!");          } // end if          Celula tmp = primeiro;          primeiro = primeiro.prox;          int resp = tmp.elemento;          tmp.prox = null;          tmp = null;          return resp;      } // end remover()  } |

**Exercício – Slide 40:**

Implemente a pilha flexível com nó cabeça.

|  |
| --- |
| public class PilhaComCabeca{      private Celula topo;      public PilhaComCabeca(){          topo = new Celula();          topo.prox = null;      } // end Pilha()        public void inserir(int value){          Celula tmp = new Celula(value);          tmp.prox = topo.prox;          topo.prox = tmp;          tmp = null;      } // end inserir()      public int remover() throws Exception{          if(topo.prox == null) throw new Exception("Erro!");          int elemento = topo.prox.elemento;          Celula tmp = topo.prox;          topo = topo.prox.prox;          tmp.prox = null;          tmp = null;          return elemento;      } // end remover()      public void mostrar() {          System.out.print("[ ");          for (Celula i = topo.prox; i != null; i = i.prox) {              System.out.print(i.elemento + " ");          } // end for          System.out.println("] ");      } // end mostrar()  } // end class PilhaComCabeca |