UDF Centro Universitário

Estrutura de Dados

Pilhas

Prof. Antonio PEREIRA

Cruzeiro do Sul Educacional

Lista Linear

- Uma lista linear é um tipo abstrato de dado (TAD) que representa uma coleção L:[a_1,a_2,...,a_n], n>=0 cuja propriedade estrutural baseia-se apenas na posição relativa dos elementos, que são dispostos linearmente.
- Se n=0, dizemos que a lista L é vazia; caso contrário, são válidas as seguintes propriedades
  - a_1 é o primeiro elemento da lista L;
  - a_n é o último elemento da lista L;
  - a_k, 1<k<n, é precedido pelo elemento a_{k-1} e seguido por a_{k+1} na lista L.

Pilha

- A Pilha é um tipo especial de lista linear em que todas as inserções e remoções são feitas em uma mesma extremidade denominada topo. Então, temos acesso somente ao topo da pilha, ou seja, quando queremos inserir um elemento na pilha, colocamos no topo e, se quisermos excluir um elemento da pilha, só podemos excluir aquele que está no topo.
- Uma Pilha possui três operações básicas:
  - top(): acessa o elemento posicionado no topo da pilha e o retorna;
  - push(elemento): insere um novo elemento no topo da pilha;
  - pop(): remove um elemento do topo da pilha e o retorna;

Pilha

--------	P:[ ]	Inicialmente a Pilha está Vazia.
push(a)	P:[a]	Insere o elemento a na Pilha P.
push(b)	P:[a, b]	Insere o elemento b na Pilha P. Note que, agora, o topo é b.
push(c)	P:[a, b, c]	Insere o elemento c na Pilha P. Agora o topo contém c.
pop()	P:[a,b]	Remove o elemento do topo e retorna o conteúdo. No caso c.
push(d)	P:[a, b, d]	Insere o elemento d no topo..
top()	P:[a, b, d]	Não altera a Pilha, apenas retorna o valor do topo da Pilha (d).
push(top())	P:[a, b, d, d]	Insere no topo mais uma cópia do elemento que estava no topo.
push(pop())	P:[a, b, d, d]	Insere no topo o que foi excluído do topo (fica do mesmo jeito).

Pilha

- Além das operações básicas temos as seguintes operações:
  - vazia() – retorna se a pilha está vazia
  - tamanho() – retorna a quantidade de elementos na pilha

Como implementar?

- Abordagem estática-sequencial (vetor)
  - A pilha será implementada pela conjugação de:
    - um atributo **areaPilha** do tipo vetor
    - e um atributo ***topo*** de tipo inteiro (que armazenará o índice correspondente ao topo da pilha no vetor)
  - As operações irão utilizar o atributo topo, atualizando seu valor quando necessário.
  - Nesta abordagem ao criar a pilha precisaremos definir sua capacidade de armazenamento (correspondendo ao comprimento do vetor P).

Pilha – usando vetor

```
public class Pilha {
private final int CAPACIDADEDEFAULT = 30;
private int topo;
private Object areaPilha[];

public Pilha() {
   areaPilha = new Object[CAPACIDADEDEFAULT];
   topo = -1;
}

public Pilha(int capacidade) {
    areaPilha = new Object[capacidade];
    topo = -1;
}

public boolean vazia() {
    return (topo < 0);
}

public int tamanho() {
     return topo + 1;
}
```

```
public Object top() {
     if (this.vazia())
          return null;
     else
          return this.areaPilha[this.topo];
}

public void push(Object elemento) {
          if (this.topo < this.areaPilha.length -1) {
                this.areaPilha[++topo] = elemento;
           }
}

public Object pop() {
     if (this.vazia())
           return null;
        else
           return this.areaPilha[this.topo--];
}

}
```

Pilha – classe de teste

```
public class TestaPilha {

public static void main(String[] args) {
      Pilha P1 = new Pilha(20);

     P1.push("A");
     System.out.println(P1.top());

      P1.push("B");
      System.out.println(P1.top());

      System.out.println(P1.pop());

      System.out.println(P1.top());

      System.out.println(P1.pop());

     System.out.println(P1.top());
}

}
```

- Saída

  A

  B

  B

  A

  A

  null

Como implementar?

- Abordagem dinâmica (ponteiros/referência)
  - A pilha será implementada utilizando-se o encadeamento de nós criados dinamicamente (lista simplesmente encadeada)

Pilha – Lista encadeada

```
public class Nodo {
    private Object dado;
    private Nodo prox;

    public Nodo(Object dado) {
        this.dado = dado;
        this.prox = null;
    }

    public Object getDado() {
         return this.dado;
    }

    public Nodo getProx () {
         return this.prox ;
     }

    public void setProx (Nodo proximo) {
          this.prox = proximo;
    }
}
```

Pilha – Lista encadeada

```
public class Pilha {
   private Nodo topo;
   private int quantElementos;

  public Pilha( ) {
     topo = null;
    quantElementos = 0;
  }

   public boolean vazia() {
      return (topo == null);
    }

   public int tamanho() {
      return this.quantElementos;
    }

   public Object top() {
       if (this.vazia())
          return null;
       else
          return this.topo.getDado();
}
```

```
   public void push(Object elemento) {
       Nodo novo = new Nodo(elemento);
       novo.setProx(topo);
       topo = novo;
       quantElementos++;
    }

    public Object pop() {
         if (this.vazia())
            return null;
        else  {
             Object elemento = topo.getDado();
             topo = topo.getProx();
             quantElementos--;
             return elemento;
         }
    }
}
```

Pilha – classe de teste

```
public class TestaPilha {

public static void main(String[] args) {
      Pilha P1 = new Pilha();

     P1.push("A");
     System.out.println(P1.top());

      P1.push("B");
      System.out.println(P1.top());

      System.out.println(P1.pop());

      System.out.println(P1.top());

      System.out.println(P1.pop());

     System.out.println(P1.top());
}

}
```

- Saída

  A

  B

  B

  A

  A

  null

Adicionando um método toString

Na classe Pilha ....

```
public String toString() {
    String temp = new String();

    if (!this.vazia()) {
      Nodo aux;
      aux = topo;
      while (aux != null){
          if (aux.getProx() != null)
              temp = ", "+ aux.getDado().toString() +
temp;
          else
               temp = aux.getDado().toString() + temp;
          aux = aux.getProx();
       }
     }

     return "P:[ " + temp + " ] ";
}
```

Usando no método main() ....

```
P1.push("A");

System.out.println(P1.toString());

P1.push("B");

System.out.println(P1.toString());

P1.push("C");

System.out.println(P1.toString());
```