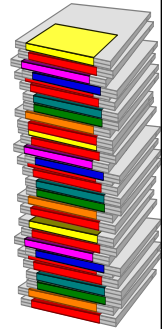


# Programação III

A Classe Pilha  
Jocélio Passos  
joceliopassos@bol.com.br

## Classe Pilha

- Last In First Out - LIFO
- Último a entrar é o primeiro a sair.
- Exemplo envolvendo os conceitos :
  - Classe
  - Objeto
  - Atributo
  - Método
  - Instância
  - Construtor
  - Encapsulamento (information hiding)
  - Sobre Carga (overloading)



2/6/2008

2

## Classe Pilha : Atributos

- `class Pilha`
- { `private int [] pilha; //Atributos encapsulados`
- `private int max;`
- `private int indice;`

2/6/2008

3

## Classe Pilha : Construtor

- `public Pilha (int max)`
- {`this.pilha = new int [max];`
- `this.max = max - 1; // Base zero`
- `this.indice = -1;`

2/6/2008

4

## Classe Pilha :Sobrecarga

- `public Pilha ()`
- {`int max = 10; // Default 10 elementos`
- `this.pilha = new int [max];`
- `this.max = max - 1; // Base zero`
- `this.indice = -1;`
- }

2/6/2008

5

## Classe Pilha : Acesso aos dados

- `public int getIndice ()`
- `// Acesso ao atributo indice`
- {`return (this.indice);`}
- `public int getMax ()`
- `// Acesso ao total de elementos empilhados`
- {`return (this.max + 1);`}

2/6/2008

6

### Classe Pilha : Métodos

```
■ public boolean pilhaVazia ()
■ {return (this.indice == -1);}

■ public boolean pilhaCheia ()
■ {return (this.indice == this.max);}
```

2/6/2008

7

### Classe Pilha : Empilhando

```
■ public boolean push (int numero)
■ // Empilha um elemento
■ { boolean ok = true;
■ if (! this.pilhaCheia ())
■ { this.pilha [ ++this.indice ] = numero; }
■ else
■ { ok = false;
■ System.out.println ("Overflow (Pilha cheia) !");
■ return (ok);}
```

2/6/2008

8

### Classe Pilha : Qual o elemento do topo ?

```
■ public int top () // Retorna o elemento do topo
■ { int numero = -1;
■ if (! this.pilhaVazia () )
■ { numero = this.pilha [this.indice];}
■ else
■ {System.out.println ("Pilha vazia !");
■ return (numero);
■ }
```

2/6/2008

9

### Classe Pilha : Desempilhando

```
■ public int pop () // Desempilha um elemento
■ { int numero;
■ numero = this.top ();
■ if (numero != - 1 )
■ { this.indice--;}
■ else
■ {System.out.println ("Underflow !");}
■ return (numero);
■ }
■ // Fim da classe
```

2/6/2008

10

### Classe Pilha : Instanciando

```
■ class TestarPilha
■ {public static void main(String[] args)
■ throws Exception
■ {Pilha pilha = new Pilha (5);
```

2/6/2008

11

### Classe Pilha : O programa

```
■ System.out.println ("Desempilha " + pilha.pop ());
■ System.out.println ("Pilha vazia ? " + pilha.pilhaVazia ());
■ System.out.println ("Empilha valor 50 " + pilha.push (50));
■ System.out.println ("Empilha valor 25 " + pilha.push (25));
■ System.out.println ("Empilha valor 32 " + pilha.push (32));
■ System.out.println ("Topo " + pilha.top ());
■ System.out.println ("Empilha valor 70 " + pilha.push (70));
■ System.out.println ("Empilha valor 5 " + pilha.push (5));
■ System.out.println ("Empilha valor 90 " + pilha.push (90));
```

2/6/2008

12

### Classe Pilha : Acessando os dados

- `System.out.println ("Índice " + pilha.getIndice ());`
- `System.out.println ("Número de elementos " + pilha.getMax ());`
- `System.out.println ("Pilha cheia ? " + pilha.pilhaCheia ());`

2/6/2008

13

### Classe Pilha : Esvaziando a pilha

- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Desempilha " + pilha.pop ());`
- `System.out.println ("Pilha vazia ? " + pilha.pilhaVazia ());`
- `JUtil.pause ();`
- `}`
- `// fim da classe`

2/6/2008

14

### Exercícios Propostos

- 1) Altere a classe Pilha criando um método que esvazie de vez toda a pilha (esvaziarPilha).
- 2) Crie novos métodos (pilhaVazia2) e (pilhaCheia2) que devolvam não mais True/False, mas Sim/Não.

2/6/2008

15

### Exercícios Propostos

- 3) Criar uma FILA encapsulada.
- FIFO (First In First Out)
- Primeiro elemento a entrar é o primeiro a sair.

2/6/2008

16

### Exercícios Propostos

- 4) Exemplo usando os conceitos até construtor e encapsulamento crie uma classe Telefone (inteligente).  
Funcionamento :
  - Discar ao receber um numero.
  - Tocar (ao ser acionado dispara um som)
  - Armazenar o último número discado.
  - Armazenar o numero da última ligação recebida.
  - Rediscar o último numero.
  - Limpar último numero discado

2/6/2008

17

### Exercícios Propostos

- 5) Altere o exemplo anterior para que, além das funções acima, faça também:
  - Armazenar vários números discados.
  - Armazenar vários números de ligações recebidas.
  - Limpar todos os números discados.
  - Permitir que o usuário escolha um numero armazenado e disque.
  - Permitir armazenar além do número o nome da pessoa (agenda).
  - Discar um número recebendo apenas o nome da pessoa.

2/6/2008

18