

UFPI - CCN - DC
Ciência da Computação
Estrutura de Dados

Pilha



Prof. Raimundo Moura
rsm@ufpi.edu.br

33

Tipos Abstratos de Dados (ADTs)

- ▶ Do inglês: *ADT = abstract data type*.
- ▶ Um ADT é um conjunto de coisas, que chamaremos *itens*, e um conjunto *operações* sobre essas coisas.
- ▶ As operações são especificadas em uma *interface* que chamamos API (applications programming interface). A interface diz o que cada operação faz, sem dizer como faz.

34

Tipos Abstratos de Dados (ADTs)

- ▶ Exemplo de ADT: [pilha](#).
- ▶ As operações são: *inserção* de um item e *remoção* de um item (de acordo com certas regras).
- ▶ Outros exemplos de ADTs: [fila](#), [string](#), [fila priorizada](#), [tabela de símbolos](#), [grafo](#), etc.

35

Tipos Abstratos de Dados (ADTs)

- ▶ Um ADT pode ser *implementado* (ou seja, concretamente realizado) de diversas maneiras. (Uma pilha, por exemplo, pode ser implementada em um vetor ou em uma lista ligada)
- ▶ Cada implementação tem suas vantagens e desvantagens (por exemplo, algumas operações ficam mais rápidas enquanto outras ficam mais lentas)
- ▶ É praticamente impossível fazer uma implementação universal, boa para todos os usuários.

36

Tipos Abstratos de Dados (ADTs)

- ▶ Um usuário do ADT é chamado *cliente*.
Um cliente só deve usar as operações especificadas na API e não deve ter acesso aos detalhes da implementação da ADT.
- ▶ No esquema *cliente/interface/implementação*, a interface funciona como um contrato entre o cliente e a implementação.

37

Implementações de ADTs

- ▶ Em Java, cada ADT é implementado por uma *classe*. (Veja [Class \(computer programming\)](#) na Wikipedia)
- ▶ As operações do ADT são implementadas como *métodos* da classe
- ▶ Cada *instância* (ou seja, cada caso particular) de uma classe é um *objeto*
- ▶ Para criar uma instância de uma classe use o comando `new` de Java.

38

Implementações de ADTs

- ▶ **OBS:** (Nem toda classe Java é implementação de um ADT. Uma classe Java também pode ser uma biblioteca de métodos, como StdOut, ou um programa cliente. Mas não se fazem instâncias de bibliotecas nem de programas cliente)

39

Implementações de ADTs

- ▶ Um método de uma implementação de um ADT pode ser:
 - **de classe, ou estático**, se vale para todas as instâncias da classe;
 - **de instância, ou não-estático**, se cada instância da classe tem sua própria cópia do método.
- ▶ Métodos estáticos são indicados pela palavra-chave **static**.

40

Implementações de ADTs

- Exemplo: Cria uma instância s da classe Out (que implementa fluxos de saída dirigidos a arquivos) e invoca o método println() da instância:

```
Out s;  
s = new Out(nomedoarquivo);  
s.println("blablabla");
```

- As duas primeiras linhas poderiam ser juntadas:

```
Out s = new Out(nomedoarquivo);
```

41

Implementações de ADTs

- Mais um exemplo: Cria uma instância x da classe String e invoca o método equals() da instância:

```
Scanner scan = new Scanner(System.in);  
String x = scan.next();  
if (x.equals("-"))  
    System.out.println(pilha.pop());  
else pilha.push(x);
```

42

Implementações de ADTs

- Mais um exemplo: Cria uma instância de uma classe **Stopwatch** e invoca o método `elapsedTime()` da instância:

```
StopwatchCPU tm1 = new StopwatchCPU();
// trecho a ser analisado
...
System.out.println(tm1.elapsedTime());
```

43

Java: Implementações

Ambientes de Desenvolvimento (IDE - *Integrated Development Environment*)



Sublime Text



Eclipse



NetBeans 8.1.app

44

Pilha (= stack) e sua API

- ▶ Uma *pilha* (= *stack*) é um ADT que consiste em uma coleção de coisas munida de duas operações: *push*, que insere uma coisa na coleção, e *pop*, que remove a coisa mais recente da coleção. (**Portanto, a última coisa a entrar é a primeira a sair - LIFO**)
- ▶ As coisas de que uma pilha é feita serão chamadas *itens*. Os itens podem ser números, strings, structs, etc., etc.
- ▶ **Exemplos de clientes de pilhas:** botão voltar de um browser, cálculo do valor de uma expressão aritmética.

45

Pilha: API (= Interface)

- ▶ Pilha de itens (além das operações fundamentais *push* e *pop*, temos algumas operações auxiliares):

```
public class Stack<Item>
```

<code>Stack()</code>	construtor: cria uma pilha de Itens vazia
<code>void push(Item item)</code>	insere item nesta pilha
<code>Item pop()</code>	remove o Item mais recente desta pilha
<code>boolean isEmpty()</code>	esta pilha está vazia?
<code>int size()</code>	número de Itens nesta pilha

`Item` é um *tipo genérico*, ou *parâmetro de tipo*, que deve ser substituído por um *tipo concreto* quando uma *instância* da pilha é criada.

46

Pilha: Implementação

- ▶ Exemplo de cliente de pilha: lê strings da entrada padrão e as inserir numa pilha, a menos que a string lida seja "-", caso em que deve remover uma string da pilha (topo)
- ▶ Pilha implementada em vetor com redimensionamento
- ▶ Pilha implementada em lista ligada (Exercício)