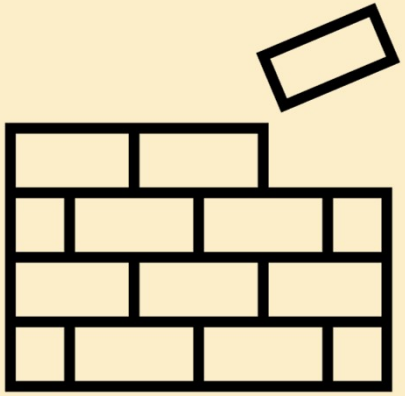
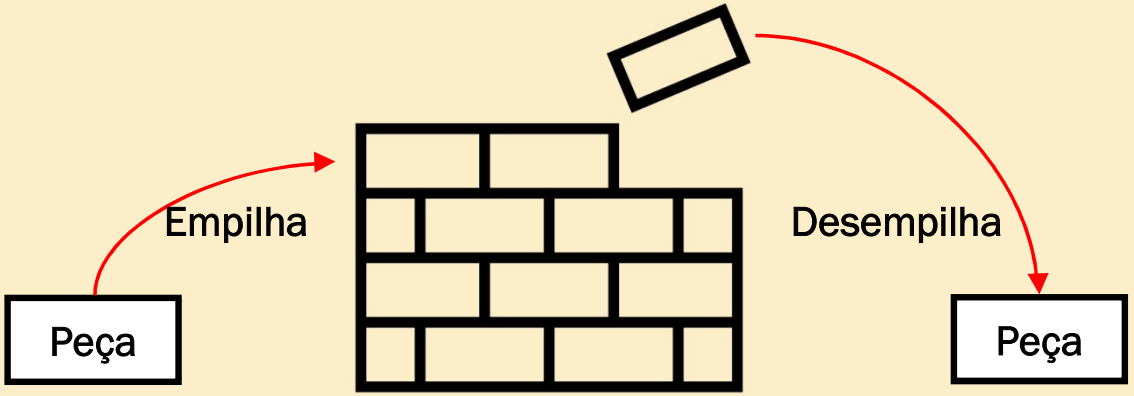


Pilhas

- Conceitos.
- Aplicação.

Pilhas

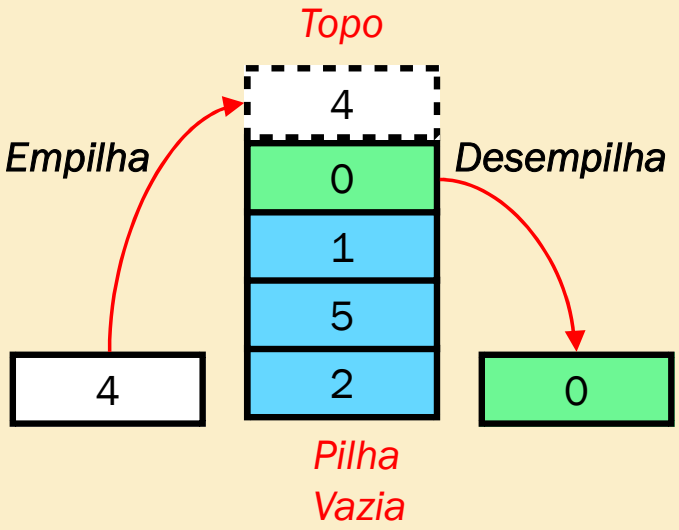
- Imagine um jogo de montar do tipo “Lego”, onde as peças devem ser montadas ou empilhadas em determinada ordem, e depois desempilhadas seguindo a ordem inversa.
- Como implementar um algoritmo capaz de realizar essa tarefa ???



Pilhas

- **Características de um Pilha**
  - Pilhas são estruturas de dados capazes de armazenar objetos de forma sequencial, similar a uma lista. Porém são mais restritivas em relação as operações permitidas, quando comparado a uma lista.
  - Permitindo o **acesso** apenas ao **último item adicionado**.
  - Conceito ***LIFO*** (***Last-in First-out***).

- **Interface de acesso a uma Pilha**
  - Uma pilha possui no mínimo 3 métodos públicos.
    - **Empilhar**: adiciona um objeto ao topo da pilha.
    - **Desempilhar**: retira o objeto do topo da pilha.
    - **Pilha Vazia**: informa que todos os objetos já foram removidos.



Aplicação de Pilhas

- **Pilhas** são essências em computação, sendo utilizadas nas mais diversas aplicações, como:
  - ***Controle de execução de instruções.***
    - Pilhas de instruções em sistemas operacionais, processadores, etc.
  - ***Compiladores.***
    - Pilhas para controle de sintática em linguagens de programação.
  - ***Interpretadores de expressões matemáticas, ortográficas e etc.***
    - Corretores ortográficos, planilhas de cálculo.
  - ***Algoritmos de busca em profundidade, como resolução de labirintos***.
    - Busca de rotas e registro de caminhos de retorno.
  - ***Controle de roteamento em redes de computadores***.
    - Controle de endereçamento de DNS, essencial para o funcionamento da internet.
  - **Etc……..**

Pilhas

- **Como implementar uma Pilha ?**
  - **Precisamos de duas variáveis:**
    - Uma estrutura capaz de armazenar objetos de forma sequência, podendo ser um vetor ou uma lista.
    - E um registrador para armazenar a posição referente ao topo da pilha.
  - **Precisamos de 3 métodos:**
    - **Empilha** que adiciona um novo item ao vetor ou lista e atualiza a posição do topo.
    - **Desempilha** que remove do topo da pilha.
    - **Pilha Vazia** que retorna verdadeiro se pilha está vazia ou falso se não estiver.

Interface pública de uma Pilha em JAVA

```
import java.util.ArrayList;
import java.util.List;

public interface InterfacePilha {

    public void empilhar(Object item);
    public Object desempilhar();
    public boolean pilhaVazia();
}
```

Em Java todos os objetos derivam da Classe ***Object***, portanto uma lista do tipo ***Object*** pode receber qual objeto.

Pilhas na API Java

- Em Java temos a Classe Stack na biblioteca ***java.util*** que implementa a estrutura de dados de uma pilha.
  - ***Exemplo:*** criando uma pilha de objetos do tipo Livro.

```
Stack<Livro> livrosParaLer = new Stack<Livro>();

livrosParaLer.push(new Livro("Data Structures and the Java Collections Framework","William J. Collins
",2011));
livrosParaLer.push(new Livro("Data Structures and Problem Solving Using Java","Mark Allen Weiss",2009));
livrosParaLer.push(new Livro("Estrutura de Dados e Técnicas de Programação","Piva D.J.; Nakamiti, G. S.;
Bianchi, F. et ",2014));

System.out.println(livrosParaLer.size());
While(!livrosParaLer.empty()){
      Livro proximo = livrosParaLer.pop();
      System.out.println(proximo.toString());
}
System.out.println(livrosParaLer.size());
```

Exercício

- Crie uma **Classe** in Java com o nome **Pilha** e implemente a ***InterfacePilha***.
- Implemente os métodos da interface para executar as funções de **empilhar**, **desempilhar** e informar que a **pilha está vazia**.
- Teste o objeto pilha empilhando uma lista sequencial de números e os desempilhando em seguida.

Utilize os métodos do objeto ArrayList() :

- **.add();** que permite adicionar itens a lista
- **.remove(index lista);** para remover itens.
- **.size();** que retorna o tamanho da lista.
- **.isEmpty();** que informa se a lista está vazia.

Interface pública de uma Pilha em JAVA

```
import java.util.ArrayList;
import java.util.List;

public interface InterfacePilha {

    public void empilhar(Object item);
     public Object desempilhar();
     public boolean pilhaVazia();
}
```

Em Java todos os objetos derivam da Classe ***Object***, portanto uma lista do tipo ***Object*** pode receber qual objeto.

Exercício

- Crie uma nova **Classe** in Java com o nome **PilhaArray**.
- Implemente os **empilhar**, **desempilhar, pilha está vazia e pilha está cheia**.
- Utilize na implementação um **Array de Object** que irá receber como parâmetro via construtor o tamanho do **Array**.
- Teste o objeto pilha empilhando uma lista sequencial de números e os desempilhando em seguida.

Utilize os métodos do objeto Array() :

- length(); que retorna o tamanho do vetor
- Crie uma variável int com nome de topo para controlar o índice do topo da pilha.
- Verifique se a pilha já está cheia antes de adicionar um novo item, e se está vazia antes de tentar remover.

Exercício

- Crie um método que recebe uma **String** e retorne **True** se a String é **Palíndromo.**
- Ou seja, o nome é lido da mesma forma nos dois sentidos.
- Para isso utilize uma pilha para inverter o nome, exemplo: “maria” ➔ “airam”. Depois compare o nome invertido com o original, se forem iguais é palíndromo.
- Utilize as classes **Pilha**, **PilhaArray** e **Stack** da API Java nos testes. E verifique se as três classes possuem o mesmo comportamento.

- Utilize os métodos do objeto String :
  - **.charAt(index);** para acessar os caracteres da String.
  - **.length():** para verificar a quantidade de caracteres da String.

Exercício

- Crie um método chamando **validaBalanceamento** que recebe uma **String** e verifica se uma sequência de caracteres contém parênteses balanceados. Retornando ***True*** ou ***False***.

- **Por exemplo:**
  - “{ [( ) ( )] }” ➔ Balanceado
  - “{ [( ))] }” ➔ Desbalanceado

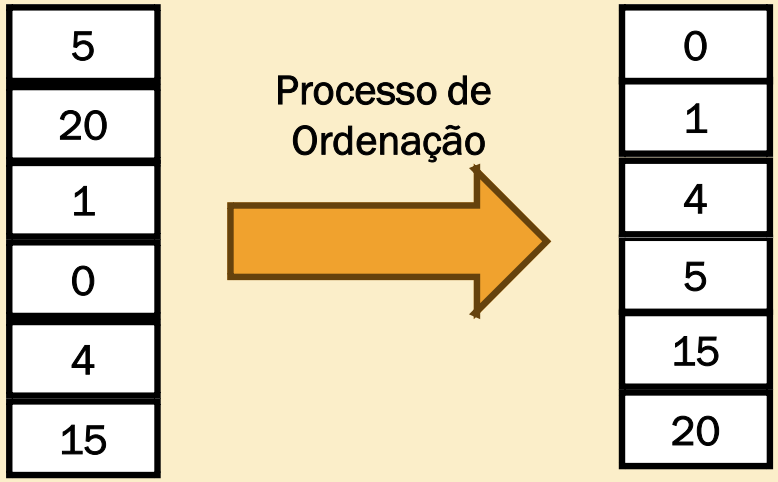
- **Utilize a seguinte lógica:**
  - Quando encontrar um símbolo de abertura **{, [, (,** adiciona o símbolo de fechamento equivalente a pilha. Por exemplo: encontra **“{“** ➔ Empilha **“}”**
  - Quando encontrar um símbolo de fechamento **},],),** compara com o topo da pilha.
    - Se for verdadeiro remove o símbolo da pilha.
    - Se for falso retorna ***False***.
  - Se o método chegar ao final da **String** e a **pilha estiver vazia**, retorna ***True***.

Exercício

- **Agora pense, em um interpretador de expressões matemáticas que precisa resolver a seguinte expressão:**
  - Resultado = [(A + B) * C] / D
- Seria possível utilizar o conceito de pilhas para solucionar esse problema ?
- Pesquise quais os passos para resolver as expressões matemáticas internas primeiro ?
- Pesquise sobre notação **Infixa**, **Prefixa** e **Posfixa**.
- Pense em utilizar funções recursivas.

Exercício

- Considere uma pilha que armazene números, escreva uma função para ordenar os valores da pilha em ordem crescente, utilize pilhas auxiliares para realizar a ordenação.
- Por exemplo:



Bibliografia

- **Estrutura de Dados e Técnicas de Programação**
  - **Piva D.J.; Nakamiti, G. S.; Bianchi, F. et (2014)**
- **Histórico da Computação e Principais Componentes Computacionais**
  - https://materialpublic.imd.ufrn.br/curso/disciplina/4/14/1
- **Estruturas de Dados em Java (Apostila)**
  - Prof. Dr. Paulo Roberto Gomes Luzzardi (2010).
- **Estruturas de Dados Abertas(Livro)**
  - Pat Morin e Joao Araujo (2021)
- **Algoritmos e Estrutura de Dados em Java (Apostila)**
  - Caelum ensino e inovação. (2021)