Pilha
Exercícios de Implementação

- **Exercício i2.1:** No projeto **Aula02**, implemente, na classe **`Exercicioi2p1`**, o método **`public static boolean isBalanced( String expression, char... pairs ) throws IllegalArgumentException`**. Esse método deve verificar se uma expressão composta por pares de símbolos arbitrários está balanceada. Uma descrição mais detalhada do método pode ser encontrada no projeto.

Pilha
Exercícios de Implementação

- **Exercício i2.2:** Expressões aritméticas podem ser escritas em três notações diferentes. Estamos acostumados à notação infixada, em que um operador é cercado por dois operandos. Por exemplo, `7 * 9`. Além da notação infixada, há também as notações pré-fixada e pós-fixada. Na notação pré-fixada, os operadores vêm antes dos operandos, por exemplo `* 7 9`, enquanto notação pós-fixada, os operadores vêm após os operandos, ou seja, `7 9 *`. Uma vantagem das notações pré e pós-fixadas é que a precedência dos operadores já é codificada na própria expressão, não havendo necessidade de, por exemplo, como na notação infixada, cercar operações por parênteses quando queremos que estas sejam calculadas primeiro. A conversão entre essas três notações pode ser feita usando pilhas! No projeto **Aula02**, você encontrará a classe `Exercicioi2p2`, que contém a descrição de todos os algoritmos de conversão. Você deve implementar os 6 métodos a seguir. Perceba também que na mesma classe, existem diversos métodos para lhe auxiliar na implementação.

Pilha
Exercícios de Implementação

- **Exercício i2.2:** Implemente os métodos:

a) `public static String prefixToInfix( String prefix ) throws IllegalArgumentException`

b) `public static String prefixToPostfix( String prefix ) throws IllegalArgumentException`

c) `public static String postfixToInfix( String postfix ) throws IllegalArgumentException`

d) `public static String postfixToPrefix( String postfix ) throws IllegalArgumentException`

e) `public static String infixToPrefix( String infix ) throws IllegalArgumentException`

f) `public static String infixToPostfix( String infix ) throws IllegalArgumentException`

Pilha
Exercícios de Implementação

- **Exercício i2.3:** No projeto **Aula02**, implemente, na classe **`Exercicioi2p3`**, o método **`public static double evaluate( String expression ) throws IllegalArgumentException`**. Esse método avalia uma expressão aritmética fornecida em qualquer forma (pré-fixada, infixada ou pós-fixada), gerando o resultado. As operações de adição, subtração, multiplicação e divisão devem ser suportadas. Uma descrição mais detalhada do método pode ser encontrada no projeto. No projeto **AlgoritmosEstruturasDeDados** há uma classe chamada **`StackBasicAlgorithms`** com a implementação do método **`evalueatePostfixExpression`** que pode ser usado como base para resolver esse exercício.

Pilha
Exercícios de Implementação

- **Observação:** Os esqueletos de todos os métodos que devem ser implementados, suas descrições detalhadas, bem como todos os testes de unidade, estão disponíveis no projeto **Aula02** fornecido no material da disciplina. Esse projeto tem como dependência o projeto **AlgoritmosEstruturasDeDados**, disponível no GitHub. Ele já foi configurado com essa dependência, então você não precisa se preocupar. Os exercícios envolvem o uso de pilhas, sendo assim, você deverá usar a implementação das pilhas que são fornecidas nesse projeto e não as versões disponíveis na sua instalação do Java Development Kit.

Pilha
Desafio ☠

- **Desafio d2.1:** Pesquise sobre o algoritmo de geometria computacional chamado Graham Scan (Exame de Graham), que tem como objetivo obter a envoltória convexa (*convex hull*) de um conjunto de pontos. Implemente na classe **`Desafiod2p1`** do projeto **Aula02** o método **`public static List<Point2D.Double> getConvexHull( List<Point2D.Double> points )`** que processa uma lista de pontos e gera a envoltória convexa desse conjunto. A classe **`Desafiod2p1`** contém um método **`main`**, que ao executado (Shift+F6) abrirá uma interface gráfica de teste para você verificar seus resultados.