**Métodos de ordenação animados**
**Proj1**

**Informações**

- Entrega pelo Portal COMP em <`http://trab.dc.unifil.br/moodle/`>;

# 1 Introdução

Neste trabalho, vamos construir um animador de algoritmos para ilustrar a execução das técnicas de busca e ordenação que aprendemos em sala de aula. Vamos representar graficamente com barras uma lista numérica qualquer, entrada pelo usuário.

O trabalho é ambicioso mas muitas das funcionalidades já estão pré-implementadas no projeto "Animador de Algoritmos", disponível no Portal COMP. Por isso, recomendo que utilize-o como ponto de partida. Nele, a interface gráfica já está implementada, bem como a funcionalidade de gravar e reproduzir o processo algorítmico.

De maneira geral, caberá ao aluno implementar:

- O desenhista de listas numéricas com variações de cores para cada elemento;
- Os algoritmos de busca e ordenação, com algumas anotações para gravação;
- Estender algumas funcionalidades do gravador, para possibilitar e facilitar o correto processo de gravação para cada algoritmo.

# 2 Roteiro de Trabalho

Para facilitar a construção desse trabalho ambicioso, separei as atividades em duas etapas principais. É necessário ter a primeira parte inteiramente pronta antes de iniciar a segunda.

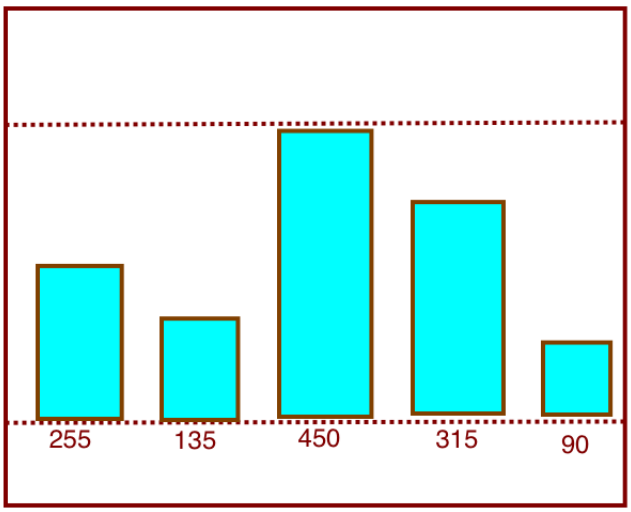
## 2.1 Desenhista de listas

Utilize o projeto "AnimadorAlgoritmos", disponível no Portal COMP, como ponto de partida para programação das funcionalidades a seguir.A princípio, trabalhe apenas com a opção "Lista estática" selecionada na ComboBox de algoritmos. A programação do desenho é feita no método `pintar` da classe `ListaGravada`. É permitido, e até desejável, criar novos métodos e classes auxiliares para fazer o desenho.

1. Tente criar um desenho qualquer neste método e confirme que ele figura na tela do programa após você preencher o campo "Valores" e apertar o botão "Carregar".

2. Dada uma lista de entrada qualquer, desenhe colunas (retângulos) que representem seus 5 primeiros elementos. O elemento de maior tamanho deverá ser representado por uma coluna com altura de 300 pixels, enquanto os outros elementos serão uma fração proporcional do tamanho desta coluna[1]. Por exemplo:

   > *Considere a seguinte lista: { 225, 135, 450, 315, 90 }. Como o maior elemento é 450, ele será representado pela coluna de 300 pixels de altura. Proporcionalmente, a coluna do valor 225 terá 150 pixels, pois 225 é metade de 450. Prosseguindo, 135 terá 90 pixels, 315 terá 210 pixels e 90 terá 60 pixels. A imagem a seguir ilustra o caso:*




3. Cada coluna deverá ter 72 pixels de largura, e deverá estar 50 pixels afastada das colunas adjacentes. As colunas de fora deverão respeitar uma margem de 20 pixels em relação às bordas da tela.

4. As colunas deverão ser coloridas todas com a mesma cor e estar delineadas com a cor preta.

5. Escreva abaixo de cada coluna o valor que ela representa, como mostrado no esboço.

6. Agora que você já dominou o desenho de uma lista com 5 elementos, está na hora de generalizar para *n* elementos. Adapte a programação que você fez para calcular quantos elementos há na lista, qual a largura deverá ter cada coluna e quanto de espaço deverá ter entre elas, de tal forma que todas caiba na tela do desenhista, independente do tamanho da tela e da quantidade de elementos na lista.

## 2.2 Ordenações animadas

7. No método `paintComponent` da classe `Tocador`, há duas funcionalidades desejadas, mas ainda não implementadas e descritas em comentários, para exibir certas informações na tela. Implemente-as.

[1] Este tipo de proporcionalidade é conhecido como *normalização*.

8. Na classe `ListaGravada`, implemente o método `pintar`. Ele é muito similar ao que você fez na seção 2.1, por isso utilize o seu código como base. `ListaGravada` possui como atributos uma lista de valores e outro de cores.

   A lista de valores deverá ser desenhada como retângulos coloridos e com borda preta, sendo que a cor de preenchimento de cada retângulo é indicado pela cor de mesma posição na lista de cores. Ou seja, o retângulo que representa o elemento `lista.get(i)` deve ser preenchido com a cor `coresIndices.get(i)`. Se a cor do elemento da lista for `null`, utilize como padrão a cor `Color.BLUE`.

   Após implementar corretamente esta funcionalidade, o aplicativo será capaz de exibir corretamente o processo de busca sequencial, portanto utilize-a como teste para saber se está desenhando corretamente cada quadro.

9. Na classe `AlgoritmosAnimados`, escreva um comentário em código para cada linha do método `buscaSequencial`, dando atenção especial às que envolvem uso do objeto `Gravador anim`. O objetivo é que você compreenda como utilizar o gravador para registrar o funcionamento dos algoritmos que você vai implementar a seguir.

10. Na classe `AlgoritmosAnimados`, implemente a ordenação pelo algoritmo da bolha no método homônimo. Após implementá-lo, utilize o `Gravador` para marcar os pontos de gravação do método, que são:

    - Disposição inicial da lista;
    - Cada vez que dois elementos são comparados, registre a operação, com o método adequado do `Gravador`;
    - Cada vez que ocorrer uma troca de posições entre elementos, registre-a;
    - Disposição final da lista;

11. Na classe `Gravador`, faça a documentação no formato *javadocs* para todos os seus métodos. O objetivo é que você entenda o seu funcionamento, para ser capaz de escrever novos métodos de gravação quando for necessário.

12. Na classe `AlgoritmosAnimados`, crie e implemente o método `ordenarPorSelecao`, com as devidas marcações com o objeto de `Gravação`.

    Após terminar a implementação, é necessário ligá-la na interface gráfica. Para tal, crie uma entrada para ela dentro do `switch` do método `onBtnCarregaPressionado` na classe `Animador`. Para todas as outras implementações subsequentes, será necessário fazer o mesmo.

13. Implemente, em `AlgoritmosAnimados`, o método `buscaBinária`. Os pontos de gravação a serem marcados são:

    - Disposição inicial;
    - Cada vez que o elemento for buscado em uma posição. Neste caso, deverá marcar também, com cores distintas, a posição inicial e final de busca binária da lista.

- Disposição final, com elemento encontrado destacado, caso houver;

Para poder fazer as marcações de gravação da busca binária, será necessário implementar um novo método de gravação na classe `Gravador`, que receba três índices, e os marque com cores distintas.

14. Implemente o restante dos algoritmos animados:
    (a) Ordenação por Inserção: deverá marcar as comparações, os deslocamentos à direita e as inserções.
    (b) Ordenação Mergesort: deverá marcar as subdivisões, comparações e intercalação.
    (c) Ordenação Quicksort: deverá marcar as subdivisões, comparações e trocas de posição. Marcar distintamente (utilizando outra cor) quando a troca for com o pivô.