Atividade - 1
Implementação de Métodos no ProOF

Seguindo os passos abaixo, as explicações dadas em sala de aula e as instruções disponíveis no tutorial do ProOF, insira os métodos descritos nos Algoritmos 1 e 2.

1º Passo: criar uma nova classe especializada a partir da classe MetaHeuristic.

2º Passo: definir um nome para o método especializando a função membro name.

3º Passo: especializar a função membro services herdada de MetaHeuristic e nela realizar a vinculação do problema e dos operadores para o método.

4º Passo: definir os parâmetros do método especializando a função membro parameters herdada de MetaHeuristic.

5º Passo: implementar o código do método dentro da função membro execute herdada de MetaHeuristic.

6º Passo: modificar a classe do tipo Factory (fRun) para adicionar o novo método ao conjunto de métodos do ambiente.

Algoritmo 1 - Algoritmo Genético Geracional

```
     /*Aloca memária e inicia aleatoriamente*/
      inicia a população com tamanho popSize
      avalia a população

    /*Apenas aloca memória, NÃO iniciando aleatoriamente*/
     inicia a populaçãoAuxiliar com tamanho popSize

      Enquanto (critério de parada não é atingido ){
       contaTamPop ← 0;//Conta as inserções em populaçãoAuxiliar
       Enquanto(contaTamPop < popSize){
            seleciona pai1 e pai2 /*Dica: use torneio do ProOF */
            filho ← crossover(pai1, pai2)
            mutação(filho)
            avaliação(filho)

            /*inserir filho na populaçãoAuxiliar*/
            populaçãoAuxiliar[contaTamPop]←filho;

            contaTamPop ←  contaTamPop + 1;
       }
      Copia populaçãoAuxiliar para população;
      }
Fim
```

Algoritmo 2 - Estratégia Evolutiva

```
     /*Aloca memária e inicia aleatoriamente*/
      inicia a população com tamanho popSize
      avalia a população

    /*Apenas aloca memória, NÃO iniciando aleatoriamente*/
     inicia a populaçãoAuxiliar com tamanho popSize

      Enquanto (critério de parada não é atingido ){

       contaTamPop ← 0;/*Conta as inserções em populaçãoAuxiliar*/

       /*Cria população auxiliar formada pelos filhos*/
       Enquanto(contaTamPop < popSize){
            filho ← população[contaTamPop]
            mutação(filho)
            avaliação(filho)
            /*inserir filho na populaçãoAuxiliar*/
            populaçãoAuxiliar[contaTamPop] ← filho;
          contaTamPop ← contaTamPop + 1;
       }

       /*Seleciona para sobrevivência a partir das duas populações*/
       contaTamPop ← 0;/*Conta as inserções em população*/
       Enquanto(contaTamPop < popSize){
         seleciona solução 1 de população /*Dica: use torneio do PrOOF*/
         seleciona solução 2 de populaçãoAux /*Dica: use torneio do PrOOF*/
         Se (solução1 melhor solução 2)
            população[contaTamPop] ← solução 1;
         Senão população[contaTamPop] ← solução 2;
       contaTamPop ← contaTamPop + 1;
       }

      }
```