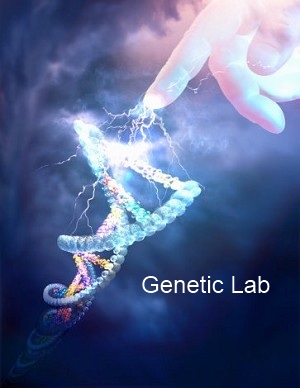
**Requisitos**



|  |  |
| --- | --- |
| Projecto: | **Life Inspiration** |
| Autor: | Genetic Lab |
| Data de preparação: | 2012-04-17 |
| Circulação: | Genetic Lab |

# Tarefas a realizar

* Fazer layout documento
* Comentar código antigo
* Testar as classes já programadas
* Programar ‘Roleta’ – Selecção
* Programar ‘SUS’ – Selecção
* Programar ‘Truncation’ – Substituição
* Programar métodos que possibilitem retirar da classe população:
* Fitness (média,…)
* Desvio Padrão
* Melhor individuo, seu fitness e atributos
* Número de indivíduos ‘com tudo a ‘1’’
* Testar
* Documentar

# Descrição sucinta das novas classes

**Classe Mochila**

|  |
| --- |
| **Mochila** |
| * ArrayList <item> items;   - Int pesoMaximo;  - Int maxPesoItem;  - Int maxValorItem;  - Int numItems; |
| + public mochila (int pesoMochila);  + public mochila (int pesoMochila, int maxPeso, int maxValor, int numItems);  + public mochila (int pesoMochila, int[] Peso, int[] Valor);  + private criaItems();  + public ArrayList<Item> getItems();  + public void setItems(ArrayList<Item> items);  + public int getPesoMaximo();  + public int getNumItems() |

Com a criação desta nova classe, a mesma irá receber os parâmetros relativos ao saco para o problema da mochila. Esta classe recebe o peso máximo da mochila e poderá receber os items que a mesma poderá conter (peso e valor de cada um) ou então calcula os mesmos aleatoriamente.

No primeiro construtor (mochila (int pesoMochila)), apenas é passado por parâmetro o peso máximo da mochila. Os items que a mesma poderá transportar bem como os seus pesos e valores serão calculados aleatoriamente.

No segundo construtor mochila (int pesoMochila, int maxPeso, int maxValor, int numItems), são passados por parâmetros o peso máximo da mochila, o máximo de peso que cada item poderá conter, o valor máximo que cada item poderá ter e o número de items que serão criados e que a mochila poderá transportar.

No terceiro construtor mochila (int pesoMochila, int[] Peso, int[] Valor), são passados por parâmetros o peso máximo que a mochila poderá transportar e os arrays de inteiros correspondentes ao peso e valor de cada item. Os items serão criados de acordo com o peso e valor passados nos arrays.

No método criarItems, este cria os items aleatórios para a mochila, passando por parâmetro para cada item o valor máximo que o mesmo poderá ter, bem como o valor máximo do mesmo.

**Classe Item**

|  |
| --- |
| **Item** |
| - Int valor;  - Int peso;  - Random random; |
| + public item (int pesoMax , int valorMax);  + public void setValor(int valor);  + public int getValor( );  + public void setPeso(int peso);  + public int getPeso( ); |
| A |

Com a classe item, irá guardar a informação referente a cada item da mochila.

O construtor desta classe é o construtor por defeito em que são passados por parâmetros o peso máximo da mochila e o valor máximo de um item para que o item gerado aleatoriamente não ultrapasse esses valores.

O método setValor() permite definir o valor do item.

O método getValor() permite devolver o valor do item.

O método getPeso() permite devolver o peso do item.

O método setPeso() permite definir o peso do item

**Classe KnapSack**

|  |
| --- |
| **KnapSack <<Individual>>** |
| - Mochila mocha; |
| + public KnapSack (KnapSack knap);  +public Mochila getMocha();  + public void setMocha(Mochila mocha); |

O construtor desta classe recebe um indiviou do tipo KnapSack e instancia o mesmo.

O método getMocha() permite devolver a mochila.

O método setMocha() permite definir a mochila.

**Classe Mochila (Exemplos de utilização)**

|  |
| --- |
| **Mochila** |
| * ArrayList <item> items;   - Int pesoMaximo;  - Int maxPesoItem;  - Int maxValorItem;  - Int numItems; |
| + public mochila (int pesoMochila);  + public mochila (int pesoMochila, int maxPeso, int maxValor, int numItems);  + public mochila (int pesoMochila, int[] Peso, int[] Valor);  + private criaItems();  + public ArrayList<Item> getItems();  + public void setItems(ArrayList<Item> items);  + public int getPesoMaximo();  + public int getNumItems();  + public boolean isPenalizacao();  + public void setPenalizacao(Boolean penalizacao); |

A mochila contém um Array de itens, de definem o peso e o valor de cada item.

O item é composto por uma valor e um peso

A mochila é definida por um peso máximo, que será o peso máximo que poderá levar.

Podemos também o máximo de peso que um item poderá ter, o máximo valor que um item poderá ter e o número de itens distintos que uma mochila poderá ter.

Na classe mochila, será também definido se os indivíduos terão ou não a penalização, aquando do cálculo do seu fitness, se os pesos que obtiverem excedam o peso máximo da mochila. Para definirmos esta penalização, utilizamos o método setPenalizacao(boolean pen). Por defeito, a penalização está activada. Se a mesma não estiver activa e caso algum individuo passe o peso máximo da mochila, o mesmo será “reparado”, sendo que será trocado, aleatoriamente, os bit’s a 1 para 0 até que o seu peso não exceda a mochila.

Exemplo de criações de mochilas:

***Mochila mocha = new Mochila (20);***

Neste exemplo, a mochila terá um peso máximo de 20, sendo que o número de itens, o máximo valor por item e o máximo de peso por item para esta mochila serão calculados aleatoriamente.

***Mochila mocha2 = new Mochila (20, 10, 30, 40);***

Neste caso, a mochila terá um peso máximo de 20, cada item terá um peso máximo de 10, o máximo de valor de cada item será de 30 e estará definido em 40 o número de itens.

***Int [] peso = {2, 4, 8, 5, 9}***

***Int [] valor = {4, 6, 1, 9, 7}***

***Mochila mocha3 = new Mochila (20, peso, valor);***

Neste caso, a mochila terá um peso máximo de 20, e os seus itens são passados por parâmetros.

Por exemplo, vai haver o item com o peso 2 e o valor 4, o item com o peso 4 e o valor 6, e assim sucessivamente.

**Classe Population – Novos métodos**

|  |
| --- |
| **Population** |
| + public double getMediaFitness();  + public double getVariancia();  + public double getDesvioPadrao(); |

O método (public double getMediaFitness()) devolve a média de fitness da população

O método (public double getVariancia()) devolve o valor da variância da população

O método (public double getDesvioPadrao()) devolve o desvio padrão da população