Programação Centrada em Objetos

Licenciatura em Tecnologias da Informação

Projeto – Fase 1

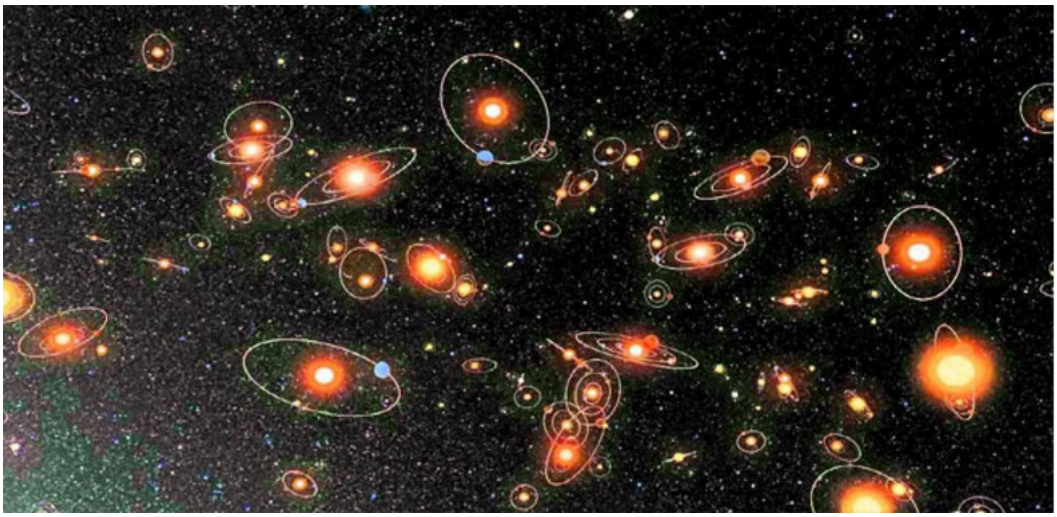
2021/2022

O objetivo final do projeto de PCO, feito em 3 fases, é pôr em prática os conhecimentos que vão sendo adquiridos nas aulas.

Nesta primeira fase do projeto vão exercitar as seguintes matérias lecionadas em PCO: declaração de variáveis, atribuição de valores a variáveis, expressões, abstração procedimental (definição de métodos *static* e sua invocação), comandos condicionais, ciclos e utilização dos tipos de dados não primitivos `String` e *array*.

Descrição do contexto onde se enquadra o objetivo deste trabalho

O seu planeta – *HomeSweetHome* – pertence a um sistema solar muito "populoso", onde existem mais de 20.000 planetas facilmente alcançáveis com a sua nave MegaSpeed X3 Turbo.



O seu grupo de amigos está a planear uma viagem de férias e já desenharam alguns possíveis trajetos de viagem que visitam vários dos planetas mais interessantes. A escolha do melhor trajeto depende de uma série de características que o grupo considera serem importantes para o sucesso da viagem. Por exemplo, de certeza que toda a gente prefere visitar planetas onde se pode respirar, para não ter que levar máscaras ou ar engarrafado.

No entanto, esta preferência tem que ser pesada a par de outras características como, por exemplo, se é possível abastecer a nave espacial durante o percurso, se a temperatura é suportável, se o ambiente é amigável, etc.

O seu grupo de amigos já pesquisou e colectou informação sobre os planetas que gostaria de visitar. De modo a poder escolher entre vários trajetos possíveis, vocês gostariam de ter uma forma fácil e rápida de perceberem se um dado trajeto tem ou não determinadas propriedades globais.

Você ficou encarregado de construir um programa que, dado um **trajeto** que percorre vários planetas, verifique se esse trajeto tem uma dada **propriedade**.

As **características** que é relevante conhecer acerca de cada planeta são representadas por termos sugestivos; por exemplo, *canBreathe, hasLight, canWalk, canRefuel, hasFood, niceSights, hasWater, niceWeather, isFriendly*, etc.

A sequência de informações sobre um planeta é expressa na forma $caract_1,...,caract_m$ onde $caract_i$ são *strings* que exprimem características. Quando m é zero, significa que não existe informação disponível sobre o planeta em questão.

Como exemplo, "*hasLight,hasFood,canWalk*" caracteriza um planeta onde existe luz natural, que tem comida disponível e onde podemos andar à vontade.

Um **trajeto** é caracterizado por uma sequência de informações acerca dos planetas que compõem o trajeto. A ordem pela qual os planetas estão descritos no trajeto é importante, pois define a ordem pela qual vão ser visitados.

Como exemplo, o seguinte *array* de *strings* define um trajeto que passa por 6 planetas, cada um com as características descritas por cada uma das *strings* que compõem o *array*, por ordem:

```
String [] percurso = {
                  "isFriendly,hasWater",
                  "hasLight,hasWater,isFriendly",
                  "hasWater,hasFood,isFriendly,canRefuel",
                  "hasLight,hasWater,canBreathe",
                  "canBreathe",
                  "hasWater,isFriendly,hasLight"
}
```

Uma **propriedade de trajeto** é uma expressão cuja veracidade acerca de um dado trajeto pode ser verificada.

Para exprimir uma propriedade de trajeto, usamos uma expressão da forma $k_1:prop_1;...;k_n:prop_n$ onde $k_1,...,k_n$ são inteiros e $prop_1,...,prop_n$ são sequências de características.

Uma propriedade de trajeto é avaliada para um dado planeta – o **planeta corrente** – num trajeto. O planeta corrente inicial é o primeiro do trajeto.

Cada inteiro indica qual a posição relativa, em relação ao planeta corrente, do planeta sobre o qual se deve avaliar a sequência de características seguinte.

Um exemplo de uma propriedade de trajeto:

`"0:hasWater;2:canRefuel,isFriendly;3:hasWater,hasLight"`

Para avaliar esta propriedade sobre o trajeto da página anterior, devemos proceder da seguinte forma:

- Por definição, o planeta corrente inicial é o primeiro do trajeto;
- Como o primeiro inteiro é zero, é em relação a esse planeta inicial que vamos avaliar a característica *hasWater*;
    - como esse planeta tem esta característica, continuamos;
- De seguida, o planeta corrente passa a ser aquele que está duas posições à frente no trajeto (neste caso é o que tem as características *hasWater*, *hasFood*, *isFriendly* e *canRefuel*)
    - Como o planeta corrente tem as duas características indicadas, *canRefuel* e *isFriendly*, continuamos;
- De seguida, o planeta corrente passa a ser aquele que está três posições à frente no trajeto (neste caso é o último do trajeto, cujas características são *hasWater*, *isFriendly* e *hasLight*)
    - Como o planeta corrente tem as duas características indicadas, *hasWater* e *hasLight*, continuamos;
- Como já não há mais propriedades para verificar, o processo termina concluindo-se que o trajeto em questão verifica a propriedade considerada.

Se, durante este processo, alguma das características em verificação não existisse no planeta corrente, concluir-se-ia que o trajeto não verificava a propriedade.

Por exemplo, o mesmo trajeto não verifica a seguinte propriedade:

`"1:canBreathe;4:hasWater,isFriendly"`

porque o segundo planeta do trajeto não tem a característica *canBreathe*. Logo que essa falha fosse verificada, o processo deveria parar.

Deverá também ser possível avaliar uma propriedade no sentido inverso do trajeto, ou seja, do fim para o início.

Por exemplo, a propriedade, `"1:canBreathe;4:hasWater,isFriendly"`, já é satisfeita pelo trajeto inverso:

- O planeta corrente inicial é o último do trajeto;
- Como o primeiro inteiro é 1, é em relação ao penúltimo planeta que vamos avaliar a característica *canBreathe*;
    - como esse planeta tem esta característica, continuamos;
- De seguida, o planeta corrente passa a ser aquele que está 4 posições para trás no trajeto (neste caso é o primeiro, que tem as duas características indicadas).

Se uma propriedade incluir referências a planetas para lá do fim do trajeto, deve-se considerar que o trajeto é circular e recomeçar do início. Por exemplo, considerando o mesmo trajeto anterior, a propriedade

"2:*canRefuel*;2:*canBreathe*;3:*hasLight*,*isFriendly*,*hasWater*"

é verificada, porque o terceiro planeta do trajeto tem a característica *canRefuel*, o quinto planeta tem a característic *canBreathe* e, finalmente, o segundo planeta tem as características *hasLight*, *isFriendly* e *hasWater*.

O mesmo raciocínio deve aplicar-se quando a propriedade é para ser verificada no sentido inverso do trajeto.

Por exemplo, considerando o mesmo trajeto, agora feito no sentido inverso, a propriedade

"2:*hasLight*,*canBreathe*;3:*isFriendly*;*2:canBreathe*"

é verificada, porque o quarto planeta do trajeto tem as características *hasLigh* e *canBreathe*, o primeiro planeta tem a característica *isFriendly* e, finalmente, o penúltimo planeta tem a característica *canBreathe*.

O que se pretende de vós nesta 1ª fase do projeto?

A ideia é a de construir um método `verificaPropriedade` (descrito mais à frente) que deverá decidir se um dado trajeto verifica uma dada propriedade, quando feito num dado sentido.

Para testarem o vosso método, devem incluí-lo numa classe de nome `MetodosVerificacao`, que <u>não</u> contém método `main`.

De seguida, devem executar a classe `PCOFase1` por nós fornecida. Nesta classe, o vosso método é invocado (daí a necessidade de estar guardado numa classe de nome `MetodosVerificacao`) para valores variados dos seus parâmetros e os resultados obtidos podem ser comparados com os corretos.

Descrição do método pretendido

O método que precisam de implementar nesta 1ª fase do trabalho tem a seguinte assinatura:

```
static boolean verificaPropriedade(String [] trajeto, String propriedade,
                                                        String sentido)
```

e é descrito através da seguinte documentação tipo Javadoc:

```
/**
 * Um dado trajeto, num dado sentido, satisfaz uma dada propriedade?
 * @param trajeto O trajeto em questao
 * @param propriedade A propriedade a ser verificada
 * @param sentido O sentido a considerar no trajeto para a verificacao
 * @requires trajeto != null && propriedade != null &&
 *           sentido in {"REGULAR","INVERSO"} &&
 *           os elementos de trajeto são sequencias de caracteristicas da
 *           forma caract1,...,caractm
 *           propriedade e' da forma  k1:prop1;...;kn:propn onde cada ki e'
 *           um inteiro e cada propi e' uma sequencia de caracteristicas da
 *           forma caract1,...,caractm
 */
```

Sugerimos **fortemente** a criação de outros métodos, privados, de modo a controlar a complexidade da vossa tarefa. Por exemplo, métodos para calcular qual a posição, no *array* `trajeto`, do próximo planeta corrente, para verificar se um dado planeta tem umas dadas características, etc.

O vosso método `verificaPropriedade` implementará então os passos gerais do algoritmo, invocando, quando necessário, os outros métodos para obter os valores/efeitos necessários em cada passo do algoritmo.

Notas importantes

- Quando temos uma *string* cujo conteúdo é formado por várias partes separadas por um caráter específico, podemos obter essas várias partes usando o método

  ```
  public String[] split (String regex)
  ```

  Se for "`0:`*`canBreathe`*`;2:`*`hasWater`*`,`*`isFriendly`*`;3:`*`hasLight`*" o valor da variável `s`, então, após a execução da instrução `String[] partes = s.split(";");` o vetor `partes` terá os seguintes elementos:

  "`0:`*`canBreathe`*", "`2:`*`hasWater`*`,`*`isFriendly`*" e "`3:`*`hasLight`*"

- Para obter o inteiro correspondente ao conteúdo de uma *string*, podemos usar o seguinte método da classe `Integer`:

  ```
  public static int valueOf(String s)
  ```

  A instrução `int numero = Integer.valueOf("123");` atribui à variável `numero` o valor inteiro 123.

- Para saber se uma *string* inclui outra *string*, podemos usar o seguinte método da classe `String`:

  ```
  public boolean contains(CharSequence s)
  ```

  Se for "*`canBreathe`*`,`*`hasWater`*`,`*`isFriendly`*" o valor de uma variável `planeta`, então, após a execução da instrução `boolean temProp = planeta.contains("`*`hasWater`*`");` a variável `temProp` terá o valor `true`.

O que entregar?

Não há relatório a entregar porque o vosso *software* é a vossa documentação. Assim, têm que comentar condignamente a vossa classe `MetodosVerificacao`: incluir no início da classe um cabeçalho Javadoc com `@author` (número do grupo e nome e número dos alunos que compõem o grupo); para cada método definido, incluir a documentação Javadoc apropriada.

Apresentem um programa cujo texto siga as normas de codificação em Java aprendidas nas aulas, bem alinhado e legível.

Para entregar: Um ficheiro *zip* com a classe que compõe a vossa solução e com os ficheiros Javadoc que geraram a partir dela (relembrem como fazer isto no Guião do Eclipse).

O nome do ficheiro *zip* que contém o vosso trabalho deverá ter o formato `PCOxxx.zip` (onde `xxx` é o número do vosso grupo).

Como entregar o trabalho?

Através do Moodle de PCO. Às 23h55 do dia acordado para a entrega, 27 de Outubro, os trabalhos entregues serão recolhidos.

Atenção que ao entregar o trabalho está a comprometer-se com o seguinte:

- O trabalho entregue é atribuível única e exclusivamente aos elementos que constituem o seu grupo;
- Qualquer indício de plágio será investigado e poderá levar ao não aproveitamento dos elementos do grupo e consequente processo disciplinar.