PRÁTICA LABORATORIAL 12

Objetivos:

- Herança
- Overriding
- Casting de Objetos

EXERCÍCIOS

Parte 1

1. O Instituto de Vida Mundial® é responsável por observar, estudar e armazenar informações relativas a todos os seres vivos. Com isto, foi contactado pelo IVM® de forma a criar uma pequena aplicação que refletisse o comportamento e permitisse simular a vida de determinado ser vivo num ambiente.

   Para isso, deve criar a classe SerVivo que tem como atributos:

   - String Nome (nome do animal)
   - String Espécie (espécie)
   - String País (país de origem)
   - Int Idade (idade)

   Seguidamente, crie as classes que irão herdar de SerVivo:

   - Planta, que terá os seguintes atributos:
     - Família (Árvores, Flores, Ervas, ComeInsetos)
     - Int Grau de Defesa (0 - 5) sendo 0 a inexistência de um mecanismo de defesa e 5 como defesa muito boa.
   - Animal, que terá os seguintes atributos:
     - Boolean fome
     - Double Peso (peso em Kg.)
     - Double inteligência (0 - 100)
     - Alimentacao dieta (Carnívoro, Herbívoro, Omnívoro)
     - String barulho
   - Inseto, que terá os seguintes atributos:
     - Boolean Venenoso

A seguir, crie a classe MeioAmbiente que terá os seguintes atributos:

- String nome
- Double agua (litros de água no meio)
- ArrayList<SerVivo> seres

Seguidamente, crie os métodos para a classe MeioAmbiente:

- Boolean plantaBebe(int indexPlanta): analisa a água disponível no meio ambiente e, caso seja possível, de acordo com a família: árvores bebem 1L, flores bebem 0.1L e ervas bebem 0.25L e retorna true (a água em meio ambiente deve também diminuir). Se retornar true, então a planta bebeu e água do meio diminuiu. Se retornar false, a planta não bebe e seca (deve ser retirada do meio ambiente).

- Boolean plantaComeInsetos(int indexPlanta): a planta come um inseto e retorna true. Caso não existam insetos, a planta morre de fome, é retirada do meio ambiente e a função retorna false.

- Void plantaAbanaComVento(int indexPlanta): imprime na consola “Está muito vento”.

- Void animalFazBarulho(int indexAnimal): imprime na consola o barulho do animal.

- Void animalMovimenta(int indexAnimal): imprime na consola “O (nome do animal) movimentou-se”.

- Boolean animalBebe(int indexAnimal): analisa a água disponível no meio ambiente e, caso seja possível, de acordo com o seu peso: por cada kg. o animal bebe 0.025L e retorna true (a água em meio ambiente deve também diminuir). Se retornar true, então é porque o animal bebeu e água do meio diminuiu. Se retornar false, o animal não bebe e morre de sede (deve ser retirado do meio ambiente).

(Continua na próxima página)

- Boolean animalCome(int indexAnimal):
    - Um animal só irá comer se fome estiver true (se o animal tiver fome), caso contrário deve apresentar a mensagem “O animal está de barriga cheia”.
    - Se o animal for carnívoro: irá comer outro animal ou inseto
    - Se o animal for herbívoro: irá comer uma planta
    - Se o animal for omnívoro: irá comer qualquer ser vivo.
    - Para avaliar se o animal pode comer outro ser vivo temos de ter em conta diversos parâmetros:
        - Se estiver a comer uma **planta** a inteligência do animal entra em ação para evitar o seu mecanismo de defesa. Para cada 20 de inteligência sobem a sua capacidade de evitar mecanismos de defesa, sendo que plantas com mecanismo de defesa 5 apenas podem ser comidas por animais com 95 ou mais de inteligência (Exemplos: um animal com inteligência entre 0 e 19 pode apenas comer plantas com mecanismo de defesa 0. Um animal com inteligência entre 20 e 39 pode comer plantas com mecanismo de defesa 0 e 1. Um animal com inteligência entre 80 e 94 pode comer plantas com mecanismo de defesa 4). Se o animal comer, então a planta morre e deve ser retirada do meio, o atributo fome passa para false e o método retorna true. Se o animal não comer, então o atributo fome passa para true e a função retorna false.
        - Se estiver a comer um **animal** terá de ser criado um índice de capacidade baseado na seguinte fórmula: (1*peso)+(2.5*inteligência). Seguidamente, deve comparar o índice da capacidade dos dois animais. Se ganhar o animal que atacou, então o animal que atacou come, o animal que defendeu morre e deve sair do meio e o método retorna true. Caso o animal que atacou perca, então não come e o atributo fome passa para true e o método retorna false.
        - Se estiver a comer um **inseto**: se for venenoso o animal morre, é retirado do meio ambiente e o método retorna false. Se não for venenoso então o animal come, a fome passa a false e o método retorna true.

- Void insetoChateia(): deve escolher automática e aleatoriamente 1 de 3 barulhos para imprimir na consola:
    - Bzzzz bzzzz
    - Tic tic tic tic
    - Psssssss

Crie o método para SerVivo de forma a listar todas as informações associadas ao mesmo.

Crie o método para MeioAmbiente para adicionar seres vivos, assim como listar todos os presentes.

Crie o método para MeioAmbiente chamado **simulador**:

- Tem como parâmetro o número de dias que queremos simular no meio ambiente. Meio o qual terá seres vivos de vários tipos adicionados.
- Este método deve ter uma variável: **acontecimento** que será um Random de 1 até 4.
    - Esta variável deve escolher qual a classe de SerVivo que irá ter um comportamento:
        - 1: Uma planta irá ter uma ação.
        - 2: Um animal irá ter uma ação.
        - 3: Um inseto irá ter uma ação.
        - 4: Catástrofe natural.
- Seguidamente, após ter escolhido uma classe de ser vivo para ter uma ação, deve verificar todos os animais existentes no meio dessa determinada classe e guardar os seus índices num novo array auxiliar. (usar ArrayList para facilitar imenso)
- Posteriormente, seleciona novamente de forma aleatória qual o ser vivo que terá uma ação e executa uma das ações desse ser vivo. Se for:
    - **Planta**: deve escolher aleatoriamente se invoca plantaAbanaComVento( ), plantaBebe( ) ou plantaComeInseto( ) (para o caso de ser planta come insetos).
    - **Animal**: deve escolher aleatoriamente se invoca animalFazBarulho( ), animalMovimenta( ), animalBebe( ) ou animalCome( ).
    - **Inseto**: deve invocar insetoChateia( ).
    - **Catástrofe Natural**: Deve escolher aleatóriamente 1 de 3 cenários:
        - **Seca**: Diminui a água para metade.
        - **Chuvas**: Aumenta a água para o dobro.
        - **Erupção Vulcânica**: Morrem metade das plantas e metade dos animais.

Cada dia nesta simulação tem três momentos (manhã, tarde e noite), e acontece uma ação por cada momento do dia. Ou seja, por dia, 3 seres vivos vão fazer algo.
Ao fim do dia (fim do terceiro momento) deve avaliar se algum animal se manteve com fome, caso alguma animal esteja com fome, deve invocar o método animalCome( ) para esses animais. Mas desta vez, se não tiverem sucesso com a sua refeição, morrem à fome antes do final do dia.
Ao início do novo dia deve imprimir a mensagem: Um novo dia começa no (nome do ambiente). E seguidamente todos os seres que se encontram vivos. Para além disso, todos os animais ficam automaticamente com fome.
O número de dias que decorrem deve ser de acordo com o que foi passado inicialmente como parâmetro.
Ao fim dos dias todos deve imprimir quais os seres que se mantiveram vivos, caso nenhum se encontre vivo, deve imprimir o último ser a morrer.

Bom trabalho! ☺