01234567890123456789012345

Université Paul Cézanne - Licence de Math-Info 2è année

I4 : Programmation Objet

TP n^o4 : héritage - attributs de classe

On veut faire un programme qui modélise des lombrics (appelés plus communément "vers de terre"). Le lombric se caractérise par le fait que lorsqu'on le coupe en deux parties égales, chacune va former un autre lombric en grandissant. Par ailleurs, il est hermaphrodite, ce qui lui permet de s'accoupler avec n'importe quel autre lombric adulte pour en faire naître un autre. On considérera qu'à sa naissance un lombric mesure un centimètre, qu'il est adulte quand il mesure 10 cm et qu'il peut grandir indéfiniment.

1. *Ecrivez la classe* `Lombric` *contenant les méthodes suivantes :*
- un constructeur sans paramètre,
- l'accesseur `int getLongueur()` qui retourne la longueur du lombric (en cm),
- la méthode `void grandit(int n)` qui ajoute `n` centimètres à la longueur du lombric,
- la méthode `Lombric accouplement(Lombric b)` qui retourne (fait naître) un lombric par accouplement du lombric avec le lombric `b` (il faut que les deux lombrics soient adultes sinon la méthode retourne `null`),
- la méthode `Lombric coupe()` qui coupe le lombric en deux parties égales et qui retourne une des deux moitiés du lombric.

2. Les deux modes de création de lombric (scission et accouplement) peuvent poser des problèmes en terme de diversification génétique : si un lombric s'accouple avec sa moitié, le capital génétique de l'enfant est le même que celui des parents, ce qui produit à terme une dégérescence de l'espèce. C'est pourquoi nous voulons interdire l'accouplement de deux lombrics issus du même lombric par scission.

Ecrivez la classe `LombricSain` *qui hérite de* `Lombric` *en redéfinissant les méthodes* `coupe` *et* `accouplement` *de façon à ce que la méthode* `accouplement` *retourne* `null` *si les deux lombrics sont issus du même lombric par scission.* Indice : la classe `LombricSain` peut contenir un nouvel attribut et les constructeurs associés.

3. On décide que les lombrics sont tous bien nourris en permanence et qu'il n'y a finalement pas de sens à faire grandir les lombrics indépendamment. On fera donc grandir les lombrics tous en même temps d'autant de cm. Pour cela, il faut modifier la classe `Lombric`.

Ajoutez la méthode de classe `void grandissons(int n)` *à la classe* `Lombric` *(et indiquez ce qu'il faut y changer d'autre) afin que tous les lombrics existants puissent grandir de* `n` *cm en même temps (grâce à l'appel de cette méthode de classe).*

Indiquez aussi la modification à effectuer pour que la méthode `void grandit(int n)` *ne puisse plus être utilisée pour faire grandir un seul lombric indépendamment des autres.*

4. *A partir des classes précédentes, écrire un programme qui crée un lombric, le fait grandir de 13cm, le coupe en deux pour créer un nouveau lombric, les fait grandir de 10cm, les fait s'accoupler pour créer un 3e lombric, les fait grandir de 5cm puis affiche la longueur de chacun des trois lombrics.*