Le Zoo

Correction sur Correction

Nous vous proposons de créer un petit zoo, puis de le gérer. Le zoo est constitué de plusieurs enclos et chaque enclos peut contenir plusieurs animaux. Dans un enclos tous les animaux doivent être du même type (ex : les baleines avec les baleines, les aigles avec les aigles, ...). Cependant, il doit être possible de mettre n'importe quelle espèce dans n'importe quel enclos.
Le zoo contient aussi un employé, qui sert d'interface entre le zoo et vous (l'utilisateur du programme). Au travers de cet employé vous pouvez donner à manger aux animaux, les transférer d'un enclos à un autre, nettoyer les enclos...
Ci-dessous nous vous donnons les spécications minimales du programme.
Elles sont parfois incomplètes. A vous de créer des classes, des variables d'instance, des méthodes ou des interfaces supplémentaires dès que cela semble nécessaire.

Les animaux

Au minimum, le programme doit pouvoir créer des Loups, des Tigres, des Ours, des Baleines, des Poissons, des Requins, des Aigles et des Pingouins.
Pour chaque espèce, nous devons pouvoir indiquer leur poids, taille, nom de l'espece et age. De plus des booléens seront utilisés pour déterminer si l'animal a faim, dort ou est malade.
Au niveau des méthodes, en plus des accesseurs et mutateurs habituels, chaque animal doit pouvoir manger, émettre un son, être soigné, dormir ou se réveiller. Il faut aussi une méthode pour faire une clef de hachage de toutes ses caractéristiques.
Les Tigres et les Loups doivent pouvoir vagabonder. Les animaux marins doivent pouvoir nager et les animaux volant doivent pouvoir voler.
Les mamifères doivent pouvoir mettre bas, alors que les autres animaux pondent des oeufs. La naissance du nouvel animal dépend de la durée de gestation ou d'incubation de l'espèce.

Question En utilisant une hiérarchie d'héritage et des interfaces, écrivez des classes pour chacun de ces animaux. Toutes les classes écrites doivent être testées.

Les enclos

Chaque enclos peut contenir plusieurs animaux (i.e. un tableau d'animaux).
Dans un enclos tous les animaux doivent être du même type (ex : les baleines avec les baleines, les aigles avec les aigles, ...). Cependant, il doit être possible de mettre n'importe quelle espèce dans n'importe quel enclos (sauf pour les volières

et les aquariums).
Tous les enclos possèdent les caractéristiques suivantes : un nom, une supercie, un degrès de propreté (pouvant prendre comme valeur : mauvaise, correcte, bonne), le nombre d'animaux présents et le nombre maximum d'animaux qu'il
peut contenir.
Outres les accesseurs et mutateurs, les méthodes d'un enclos doivent permettre : d'acher ses caractéristiques, d'acher les caractéristiques des animaux qu'il contient, d'ajouter et d'enlever des animaux dans l'enclos, d'entretenir l'enclos si celui-ci est vide.
On distinguera deux sous-classes d'enclos particulier : les volières et les aquariums. Une volière ne peut contenir que des animaux volants. En plus des caractéristiques communes à tous les enclos, elle possède aussi une hauteur. L'entretien de ce type d'enclos nécessite aussi la vérication du sommet de la cage.
Similairement, un aquarium ne peut contenir que des animaux aquatiques.

Un aquarium possède deux variables supplémentaires, la profondeur du bassin et la salinité de l'eau. L'entretien d'un aquarium nécessite la vérication de la salinité de l'eau et le nettoyage du bassin.

Question Ecrivez les classes correspondant aux trois types d'enclos. Toutes
les classes écrites doivent être testées.

L'employé

L'employé possède les caractéristiques suivantes : le nom de l'employé, son age, son sexe. Outre les constructeurs, accesseurs et mutateurs nécessaires, les méthodes suivantes, entre autres, doivent permettre :

1. à l'employer d'examiner un enclos, Cette méthode ache les animaux
contenus dans l'enclos, ainsi que les caractéristiques de l'enclos,
2. de nettoyer un enclos si l'enclos est sale et vide,
3. de nourrir les animaux d'un enclos lorsqu'ils ne dorment pas,
4. d'ajouter à un enclos un nouvel animal lorsque c'est possible,
5. de lever un animal d'un enclos,
6. de transférer un animal d'un enclos à un autre.

En l'employé possède une dernière méthode qui constitue l'interface avec l'utilisateur. A l'aide d'un menu, l'utilisateur doit pouvoir diriger l'employé.

Question Ecrivez la classe correspondant à l'employé. N'oubliez pas de la
tester.

Le zoo

Il reste maintenant à concevoir le zoo. Un zoo possède un nom, un employé, un nombre maximal

d'enclos nbMax et un tableau de nbMax enclos. Les méthodes d'un zoo permettent

1. d'acheter le contenu de tous les enclos,
2. et d'acher le nombre d'animaux présents dans le zoo.

En le zoo contient aussi la méthode main. Le comportement de la méthode main est le suivant. Dans une boucle while :
1. pour chaque animal du zoo, on va aléatoirement modier les valeurs des variables d'instance de cet animal (par exemple on le rend malade, on l'endort ou on l'aame).
2. pour chaque enclos, on modie aléatoirement son état de propreté, sa salinité, etc
3. enn on passe la main à l'employé (donc à vous, utilisateur) pour qu'il s'occupe du zoo.

Question Ecrivez la classe correspondant au zoo. N'oubliez pas de la tester.

EasyBatch

En cas d'utilisation de EasyBatch:

```
 public static void main(String[] args) throws Exception {
        List<AnimalCSV> productList=new ArrayList<AnimalCSV>();
    org.easybatch.core.job.Job job = new JobBuilder()
                .reader(new
FlatFileRecordReader("D:\\MyJavaProjects\\etnic2\\lanterna\\animaux.csv"))
                .mapper(new DelimitedRecordMapper(AnimalCSV.class, "id","type", "poid",
"age"))
                .filter(new MyProductFilter())
                .processor(new ProductProcessor(productList))
                .build();

        JobExecutor jobExecutor = new JobExecutor();
        JobReport report = jobExecutor.execute(job);
        jobExecutor.shutdown();



        System.out.println("job report = " + productList);
    }
```

```
import org.easybatch.core.processor.RecordProcessor;
import org.easybatch.core.record.Record;

public class ProductProcessor implements RecordProcessor<Record<AnimalCSV>, Record<AnimalCSV>>
```

```
{


    private List<AnimalCSV> productList;

public ProductProcessor(List<AnimalCSV> productList2) {
this.productList=productList2;
}

public Record<AnimalCSV> processRecord(Record<AnimalCSV> record) {
productList.add(record.getPayload());
    return record;
    }



}
```

```
<dependency>
    <groupId>org.easybatch</groupId>
    <artifactId>easybatch-core</artifactId>
    <version>5.2.0</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/org.easybatch/easybatch-flatfile -->
<dependency>
    <groupId>org.easybatch</groupId>
    <artifactId>easybatch-flatfile</artifactId>
    <version>5.2.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.easybatch/easybatch-xml -->
<dependency>
    <groupId>org.easybatch</groupId>
    <artifactId>easybatch-xml</artifactId>
    <version>5.2.0</version>
</dependency>
```

Revision #8
Created 29 October 2019 20:24:22 by Admin
Updated 15 June 2022 07:36:09 by ggpilou2