penguin colonies

We would like to deal with penguin colonies in which we would like to be able to find our penguin friends as quickly as possible. For this we want to use a data structure that can efficiently implement corresponding queries. Penguin colonies use the Collection HashSet for this.

The class HashSet <E> provides an implementation of changeable sets of objects of a type E. The implementation of HashSet is designed in such a way that with a favorably chosen hash function of the objects to be managed it can be checked whether an element is present in the set and objects (of the appropriate class) can be efficiently added to or removed from the set. The implementation of HashSet uses the hashCode () method as a hash function and equals () to test for equality in the objects to be managed.

penguins

First, the Penguin class is to be expanded to include implementations of these two methods. Objects of the Penguin class have the final attributes int birthYear, String name, Gender gender. Penguins are clearly identified by the combination of these attributes. They also have the Fish favoriteFish attribute, which is not final - because tastes can change in the course of a penguin life.

equals () No results

Implement the method equals (Object other) in Penguin, which implements an equality test. The method should therefore check whether other is a penguin with the same final attributes. Provide the correct return even if the parameter other or one (or more) of the attributes involved in the comparison is null. The non-final attribute favoriteFish should not be included in the equality check. This is important so that a hash set of penguins will work even if the favorite fish changes for some of the penguins.

hashCode () No results

Now implement a useful hash function hashCode in Penguin. Your method should fulfill the hashCode contract.
(There is no clear solution here. The public test here only uses a few values to check whether you have changed the hash function.)

penguin colonies

PenguinColony represents a penguin colony and has an instance attribute HashSet <Penguin> in which the penguins of the colony are managed. In the following subtasks you can assume for simplicity that the transferred objects are not null. As well as there are no null objects in collections and penguins have no null attributes.

Complement PenguinColony with the implementation of the following methods:

uniteColonies () No results

Implement the void uniteColonies (PenguinColony otherColony) method, which takes another penguin colony otherColony and adds the penguins to the colony to the current instance. The penguin colony otherColony should no longer contain penguins after executing the method.

splitColony () No results

Implement the PenguinColony splitColony (Predicate <? Super Penguin> pred) method. All penguins that have the pred predicate are removed from the colony and a new colony is returned that consists only of these animals that have just been removed from the colony.

findFirstFriend () No results

Implement the Penguin findFirstFriend (LinkedList <Penguin> penguinFriends) method. The method is said to return the first penguin in the penguinFriends list to be found in the colony. The method should not change the colony or the list of penguinFriends.

canFeedPenguinsWithProperty () No results

Implement the method boolean canFeedPenguinsWithProperty (Predicate <? Super Penguin> pred, Set <Fish> fishes). This method should return true if the favorite fish is contained in fishes for each penguin that is in the penguin colony and fulfills the pred predicate given. Otherwise false should be returned. (It doesn't matter whether only one penguin or several penguins have the same kind of favorite fish: if one type of fish is contained in fishes, all penguins that have this type as a favorite fish can be fed.)

computeSum () No results

Implement the method int computeSum (Function <? Super Penguin, Integer> fun). The method should apply the function represented by fun to all penguins in the colony and return the sum of the values calculated by the function.