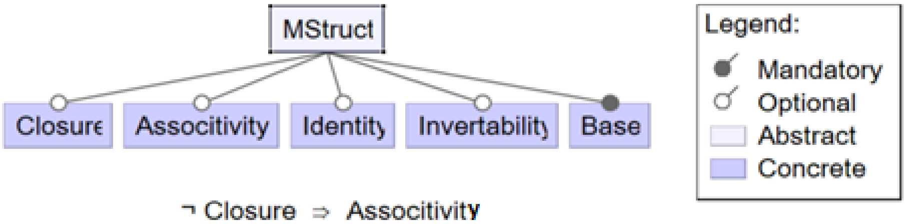
Practice Midterm

1. I created the following Feature Model (feature diagram + cross-tree constraints) from a text on Mathematical Structures. Mathematical features are Closure, Associativity, Identity, Invertibility, and Base. Of the 16 possible structures, only 9 are legal (and have specific names, which I don't show).



$$\neg \text{Closure} \Rightarrow \text{Associtivity}$$

$$\neg \text{Invertability} \wedge \text{Identity} \Rightarrow \text{Associtivity}$$

$$\neg \text{Identity} \wedge \text{Invertability} \Rightarrow \text{Closure} \wedge \neg \text{Associtivity}$$

Questions: are the following combinations legal?

a) $(closure, assoc, id, inv)$ ○ legal ○ illegal

b) b) $(\neg closure, \neg assoc, id, \neg inv)$ ○ legal ○ illegal

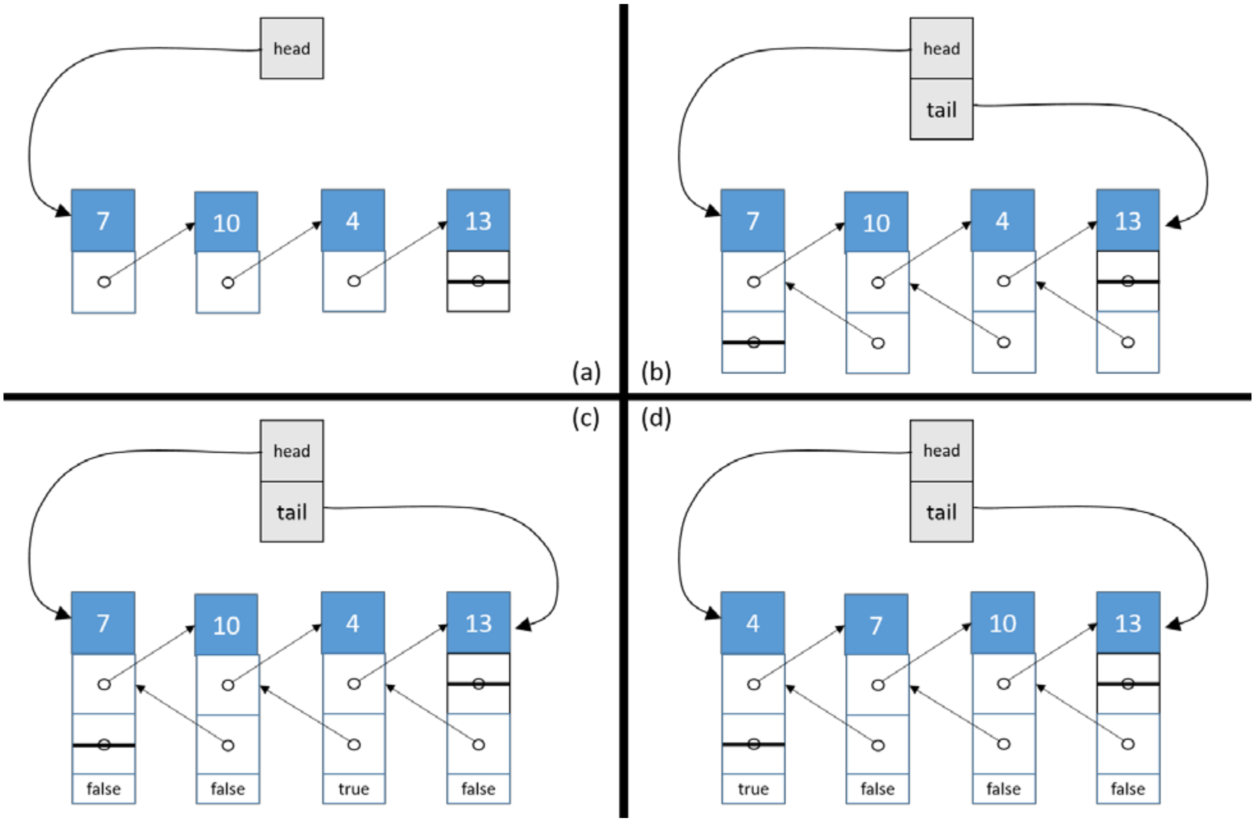
c) c) $(\ closure, assoc, id, \neg inv)$ ○ legal ○ illegal

Each of the questions below I start with no selected features. Then I select:

d) $(id$ for identity) what other assignments are selected for me?

e) $(\neg closure, id)$ what other assignments are selected for me?

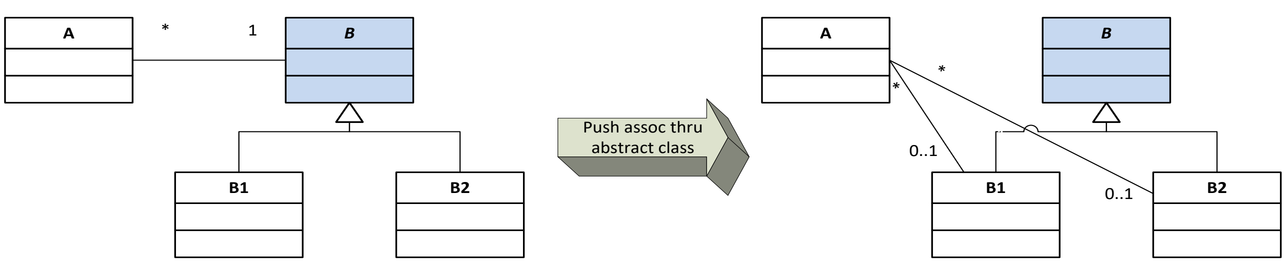
2. When I was a graduate student, it was common to find in database and data structure textbooks figures like the 4 below showing different ways of implementing lists: (a) singly-linked, (b) double-linked, (c) with delete flags (elements aren't physically removed, just marked deleted and ignored henceforth), and (d) sorted. **Note**: these figures may exhibit combinations of features, not just a single feature.



a) What is the feature model of this product line (include constraints)? You can use GuiDSl notation.
b) Shown below is a sketch of the single SPL class that implements above. "Color" this code to indicate what feature adds what fragment.

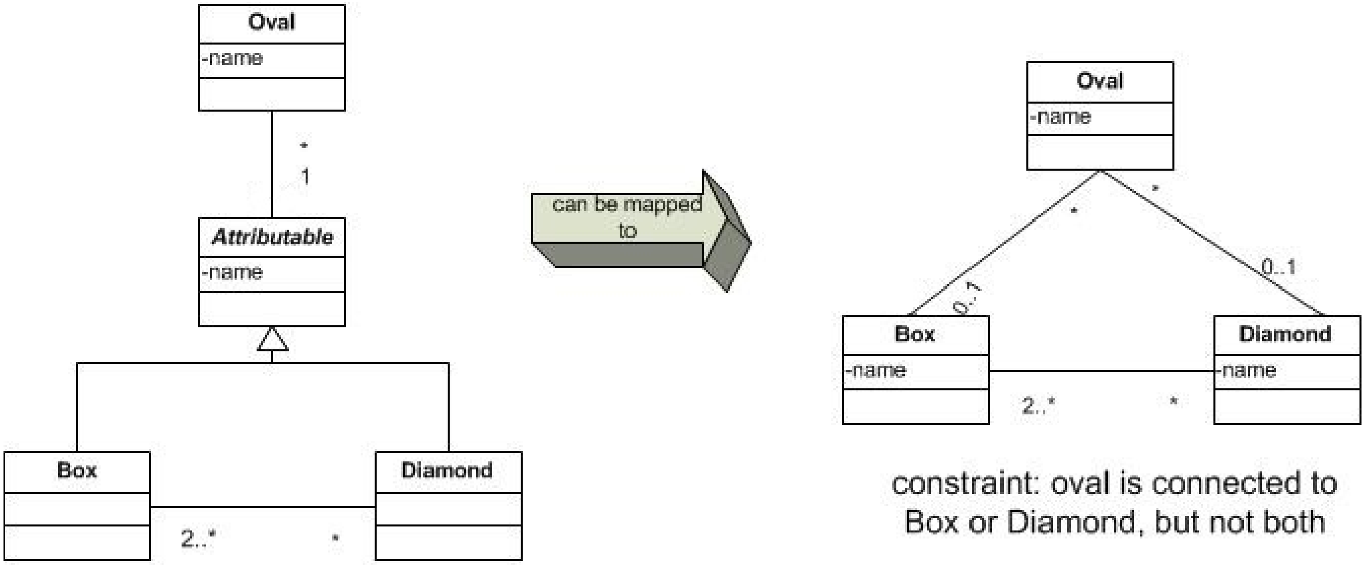
```
class MyList<R implements Comparator<R> > {

   private class ListNode<R implements Comparator<R> > {

      R value;

      ListNode next;

      ListNode prior;

      boolean deleted;

   }

   ListNode<R>  head;

   ListNode<R>  tail;

}
```

3. A common refactoring is to push an association "through" an abstract class to its subclasses:



By making class A associations reference abstract class B's subclasses, a constraint must be added: each A instance is bound to a B1 or B2 instance, but never both.

Using the above refactoring and <u>any that we have discussed in class along with their **names**</u> – show that the left model can (or cannot) be mapped to the right model.



4. Short answer

a) What is the relationship between a category and a domain-specific language? What do both represent?

b) What does an arrow of a category mean in MDE?

c) What does a composition of arrows in a category mean in MDE?

d) Give an example of a metamodel that cannot be used for bootstrapping and explain why.

Solutions

1.

a) $(closure, assoc, id, inv)$ -- group

b) $(\neg closure, \neg assoc, id, \neg inv)$ – invalid/wrong

c) ($closure, assoc, id, \neg inv)$ – monoid

d) $(ident)$ what other assignments are selected for me? (None)

e) $(\neg closure, id)$ what other assignments are selected for me (hard)? ans: only associativity,

¬closure implies associativity. so (¬closure, associativity, id) is the set of selections so far. Can we add invert?

suppose Invert is false, then ¬invert ^ identity implies Assoc, which is OK.

suppose Invert is true, then (say nothing).

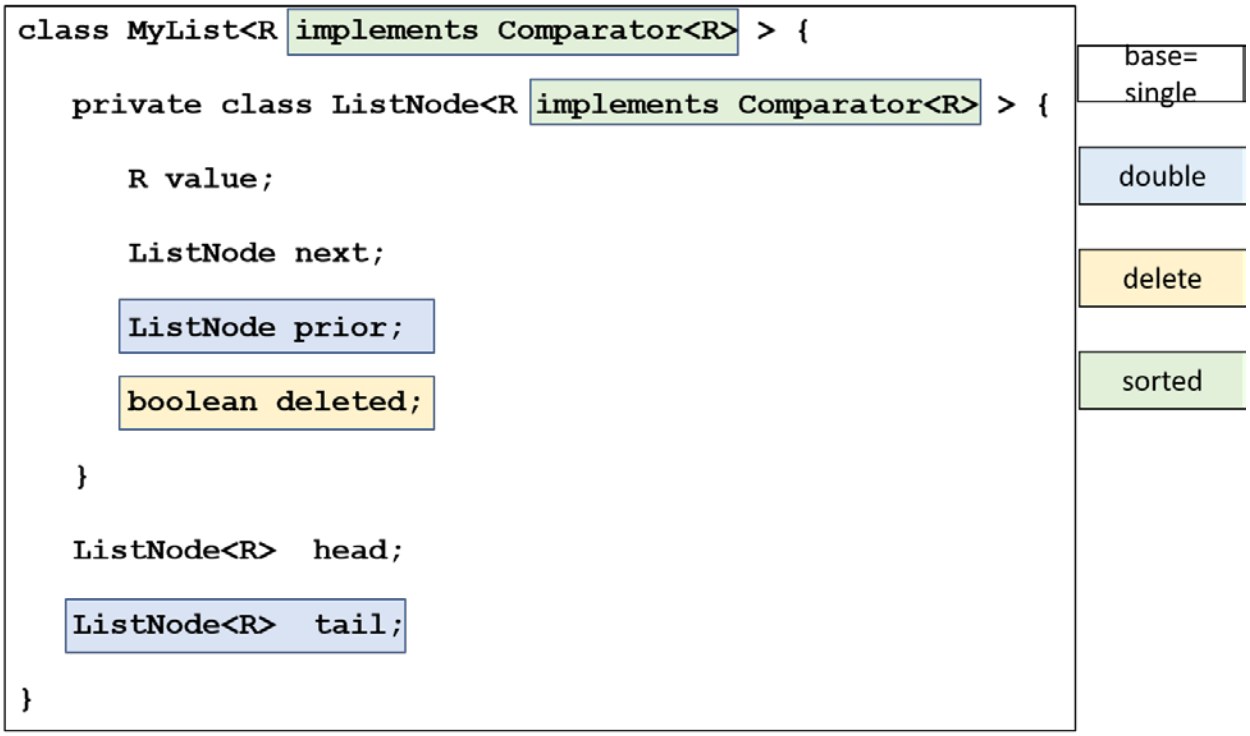
no. So only associativity is added.

2a)

MyList : [sorted] [delete] [double] base ; // base == singly-linked list

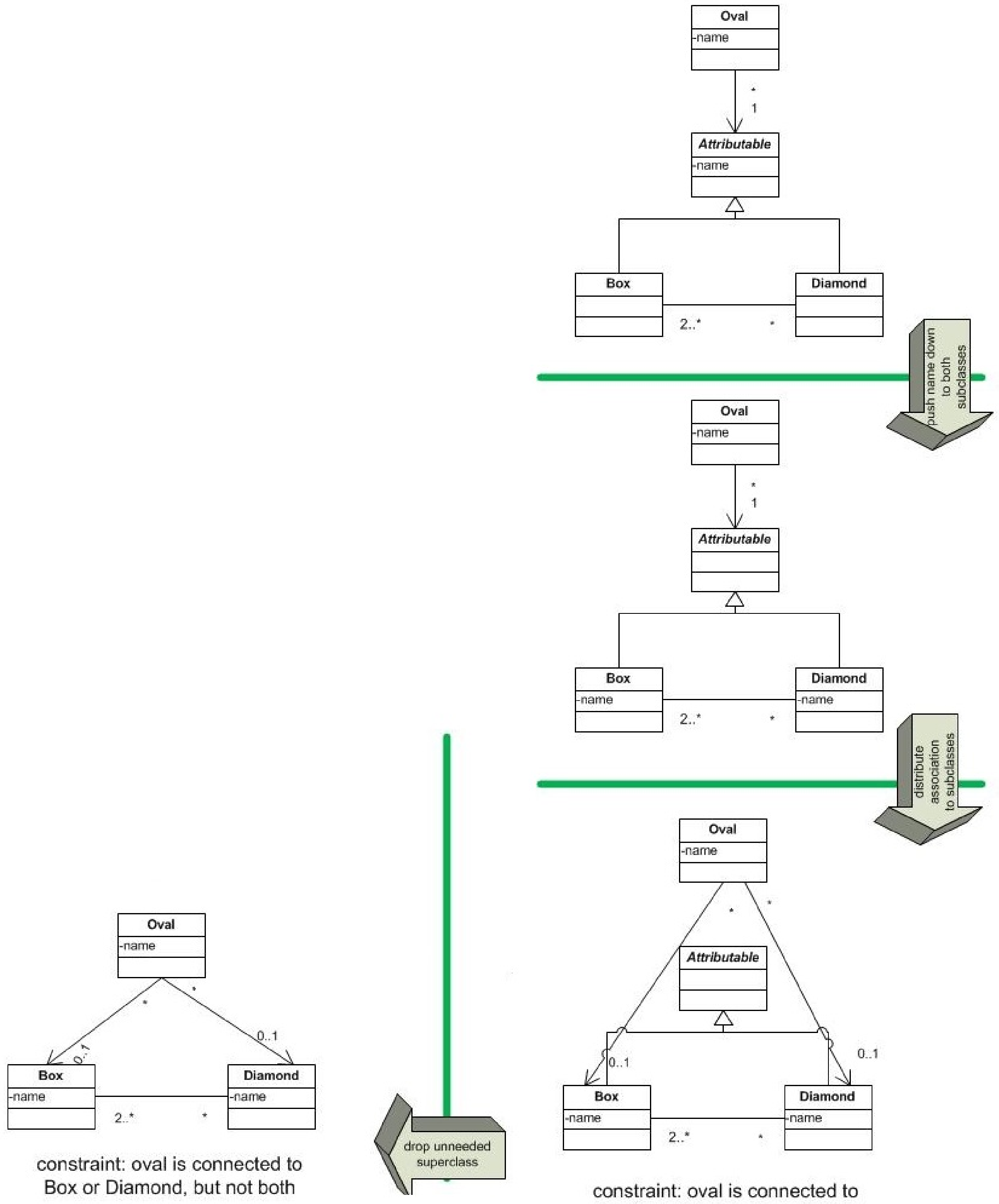
(no constraints—order of optional features is permutable)

2b)



```
class MyList<R implements Comparator<R> > {

   private class ListNode<R implements Comparator<R> > {

      R value;

      ListNode next;

      ListNode prior;

      boolean deleted;

   }

   ListNode<R>  head;

   ListNode<R>  tail;

}
```

3) Solution:



constraint: oval is connected to Box or Diamond, but not both

constraint: oval is connected to Box or Diamond, but not both

4. Short answer

a) What is the relationship between a category and a domain-specific language? What do both represent?

Ans: a category is a DSL.

both express all possible computations that can be invoked. It is a language of (legal) expressions.

b) What does an arrow of a category mean in MDE?

Ans: one possible / legal computation that can be invoked.

c) What does a composition of arrows in a category mean in MDE?

Ans: one of many possible expressions to evaluate.

d) Give an example of a metamodel that cannot be used for MDE bootstrapping and explain why.

Ans: most any metamodel can NOT be used for bootstrapping; it has to be special. Ex: a class diagram of all possible FSMs cannot be used for bootstrapping itself. It can be used to create a tool for FSMs, but not for class diagrams. A class diagram of all class diagrams CAN be used to bootstrap an MDE tools. A category diagram of a category diagram COULD be used to bootstrap an MDE tool. An ER diagram of all class diagrams could NOT be used to used to bootstrap itself.