Disjoint subclasses

This variation means that an instance can only be classified by one of the disjoint classes. Disjoint classes cannot have any overlap in their instances.

The diagram in Figure 1 shows three instances. One is an instance of “Cat”, one is an instance of “Dog”, and one is an instance of “Animal”. An instance classified as both “Cat” and “Dog” is impossible because there is no overlap between the two classes. In the most basic terms, an instance of a “Cat” cannot be an instance of a “Dog”, and vice versa.

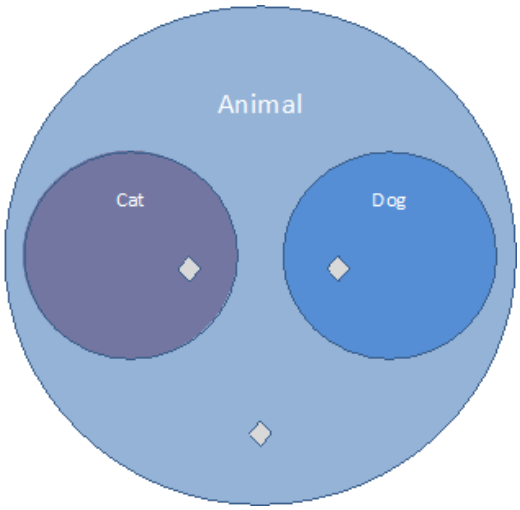

Figure 1: Disjoint instances.

The following diagram (Figure 2) shows an example of disjoint subclasses in standard UML notation. It shows that “Dog”, “Cat”, and “Mouse” are all subclasses of “Animal”. In addition, the standard UML {incomplete, disjoint} notation declares all of the subclasses to be incomplete and disjoint. Intuitively, an instance of the subclass “Dog” is an instance of the superclass “Animal”, but it cannot be an instance of the “Cat” or “Mouse” subclasses. Moreover, a lizard would be an instance of “Animal”, but could not be an instance of any of the subclasses “Dog”, “Cat”, or “Mouse”.

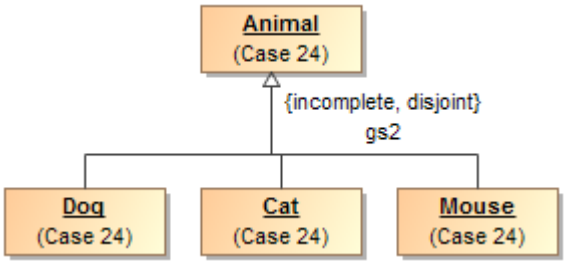

Figure 2: Incomplete and disjoint subclasses in standard UML notation.

The Concept Modeler supports importing disjoint classes. A dependency stereotyped as «Disjoint With» will be used to specify disjoint subclasses. For example, the class Animal has two disjoint subclasses, Cat and Dog. When you import them to the Concept Modeler, the diagram will look similar to the example shown in Figure 2.

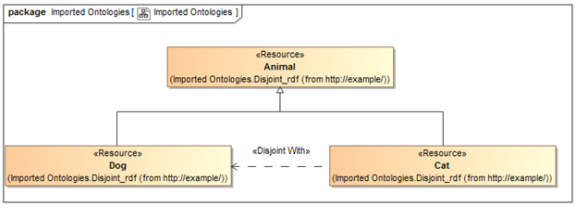

Figure 2: Imported disjoint subclasses are stereotyped with «Disjoint With».

Related pages

- Complete subclasses
- Disjoint and complete subclasses
- Overlapping and incomplete subclasses

Related Pages

Unknown macro: 'list-c