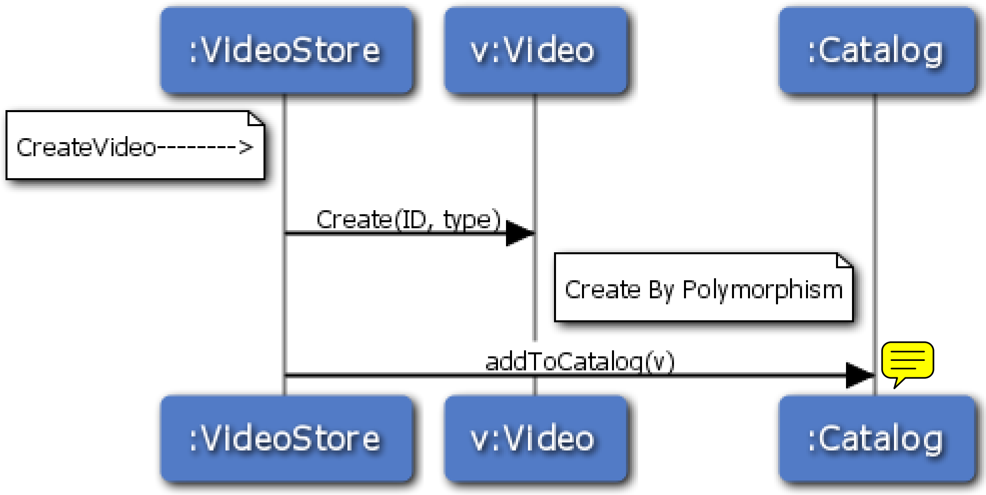
David Pick
CM 2403

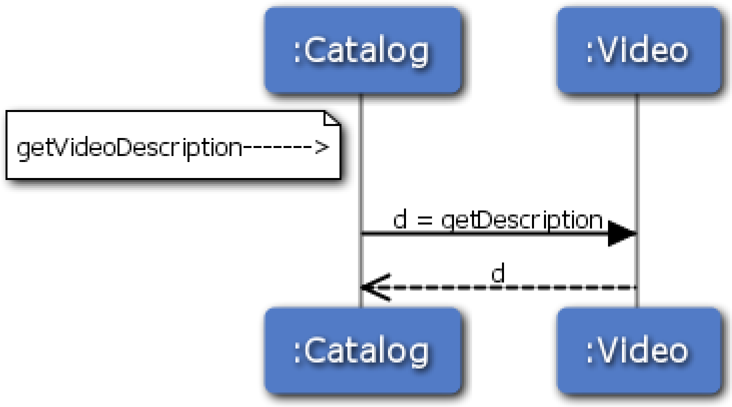
Sorry but because the tool I've been using to create class diagrams went down, I was forced to draw them by hand, and wasn't able to put the class diagrams next to the sequence diagrams, so they are labeled and are at the end of the document.

1. Add video to store / catalog - Polymorphism



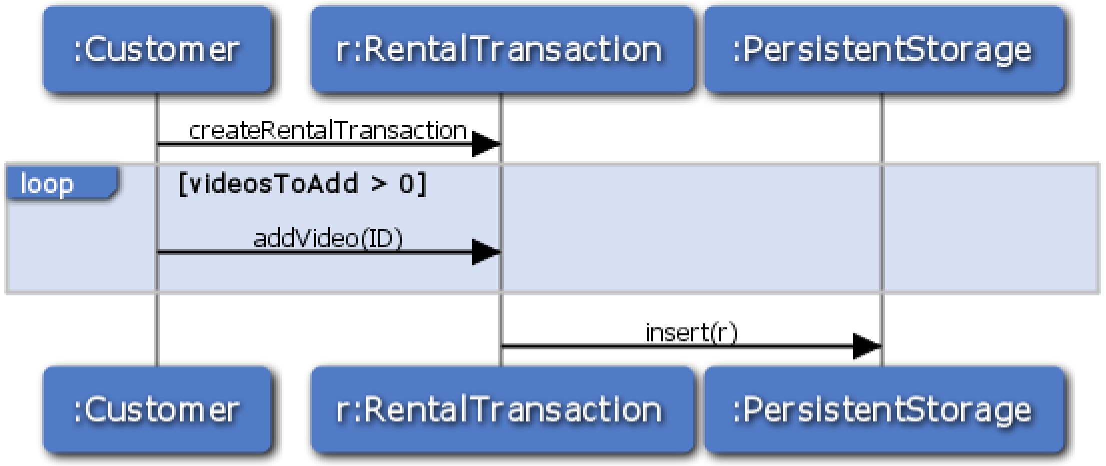
I choose to use polymorphism to create different kinds of Video objects. I choose to do this because in a video store videos are always split up by their genres. By creating these classes polymorphic-ally the system will easily be able to tell what kind of video it is dealing with, making it easier to return a subset of all videos, such as all dark comedies. This also means that if the system wants to determine if a video is a dark comedy, it doesn't first have to figure out if its a comedy, then check if its also dark, it can simply check the type of the object, instead of having to call video.genre().subgenres().

2. getVideoDescription - Indirection



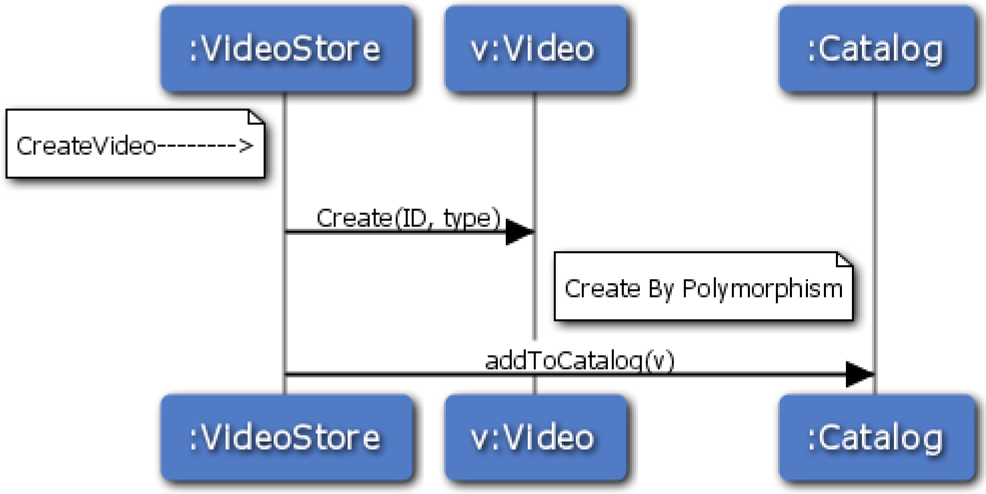
In order to avoid coupling in the VideoStore object I choose to have the Catalog by the way in which VideoStore accesses information about videos. These means that not only is VideoStore not directly connection to VideoDescription and Video, it also means that VideoStore does not need to know all the different types of Videos in the system, this can be handled by the Catalog.

3. add RentalTransaction to persistent storage - Pure Fabrication



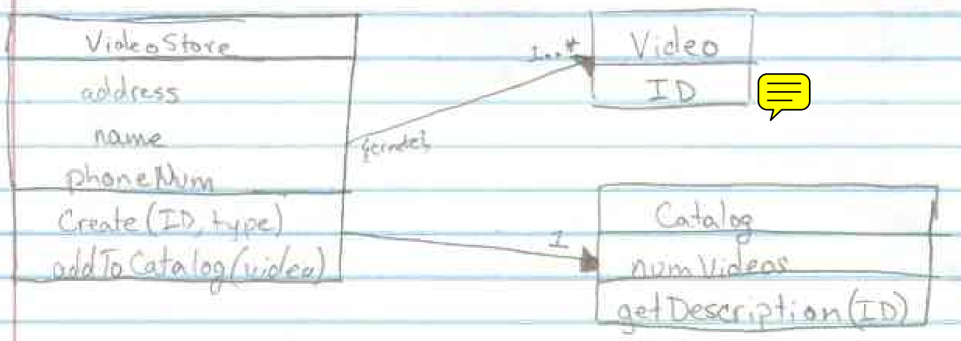
Creating something like a database object in this situtation would have caused me to violate the High Cohesion and Low Coupling goals, so I decided to create a PersistentStorage class. The PersistentStorage class will handle all interaction between the system and the database, because allowing the RentalTransaction class to directly write to the database doesn't seem like something a RentalTransaction should be able to do.

4. Add video to store / catalog - Protected Variations

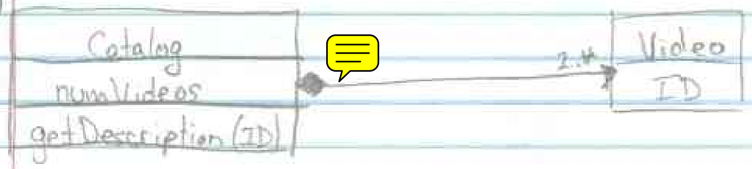


The add video to store system operation is not only an example of polymorphism, but also protected variation. Creating a new video is an example of protected variation because it is very likely that other types of videos will need to be added to the system. So by creating a superclass video which does not have a genre, and subclasses of it which each have a different genre, makes it very easy to add new kinds of videos to the system, without affecting any other parts of the system.

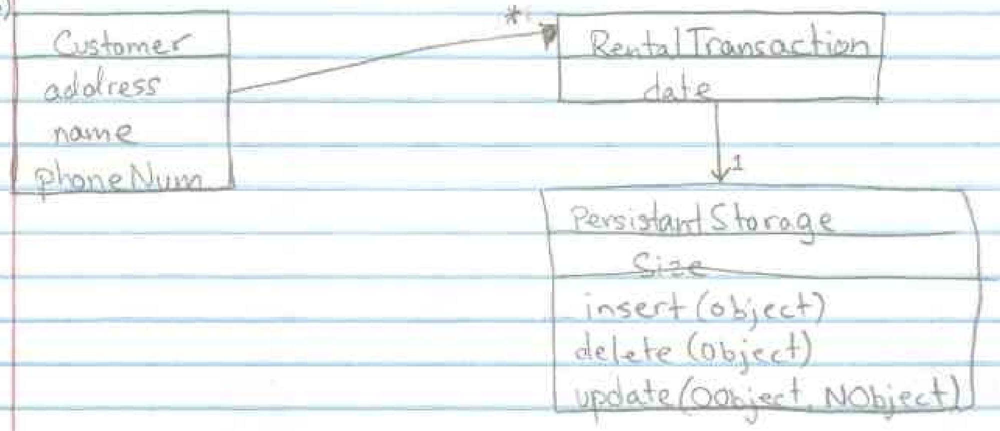
1,4)


VideoStore
address
name
phoneNum
Create (ID, type)
addToCatalog (video)
generates
1..*
Video
ID
1
Catalog
numVideos
getDescription (ID)

2)


Catalog
numVideos
getDescription (ID)
1..*
Video
ID

3)


Customer
address
name
phoneNum
*
RentalTransaction
date
1
PersistantStorage
Size
insert (object)
delete (object)
update (OObject, NObject)