BORG CALENDAR – M3

Student Name and Number

Hamid Shahrestani (9729747)  
Anjaneyulu Bodepudi (5973775)   
Manouchehr Azizi (5232287)   
Hermann Sonfack (5986052)   
Viet Hung Nguyen (9816240)

# Class Diagram of Actual System vs our Conceptual Class Diagram

To create the Actual System Model we used a Reverse Engineering tool namely “Object Aid UML Diagram” which is an Eclipse Plugin. The source code contains 18 packages, but for simplicity we only examine the core entity classes contained within “net.sf.borg.model.entity” package. The main correspondence of our modeling is with the classes contained in this package. The other 17 packages are mostly taking care of UI tasks (10 packages), tools (3 packages), database tools (2 packages), model(1 package) and one controller class (pure fabrication) which is responsible to starting up the model and spawning various threads, including the main UI thread and various timer threads. It also handles shutdown.

There’s a close correspondence between the actual system classes and those of our conceptual model. Of 15 classes we captured in our model, 8 of them exist in the actual system model. These classes are:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. CalendarEntity | 1. Address | 1. Project | 1. Task |
| 1. Memo | 1. CheckList | 1. Item | 1. Todo |

There are 7 classes in the actual system model that are not included in our conceptual model and there are 5 classes in our conceptual model that are not part of the actual system model.

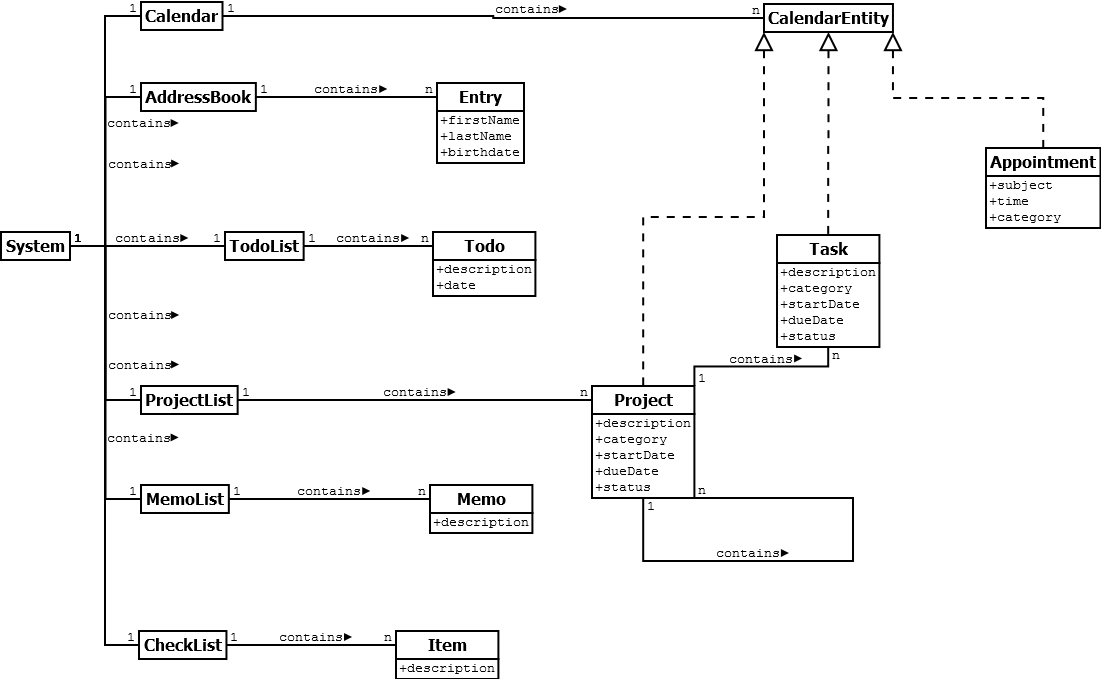
The following are the classes not included in our conceptual model:

* Option: Each BorgOption instance holds a single row from the options table in the database. This is an entity we overlooked.
* Subtask: It represent a task within another task. We figured in our design that we can model this entity with “Task” class. The use of a separate Subtask class makes the design more cohesive but instead there will be more coupling.
* Tasklog: we didn’t really take into account any logging feature but rather captured the main core functions of the software.
* Link: Any association between an entity and another entity will be represented with a Link object. This will increase coupling but might be more cohesive.
* LabelEntity: According to the documentation this class is a non-persisted entity that is used when the model needs to package a transient, calculated entity for the UI, such as a calculated holiday or birthday based on the address book. We didn’t really predicted the design that far.
* KeyedEntity: Abstract base class for all Entities that are keyed by a simple integer key. Again we considered all entities equal without defining any key for them.
* EncryptableEntity: Abstract base class for entities that can have encrypted fields. It is up to the entity specific Decrypt and Encrypt methods to determine which fields are to be encrypted. We overlooked the encryption feature.

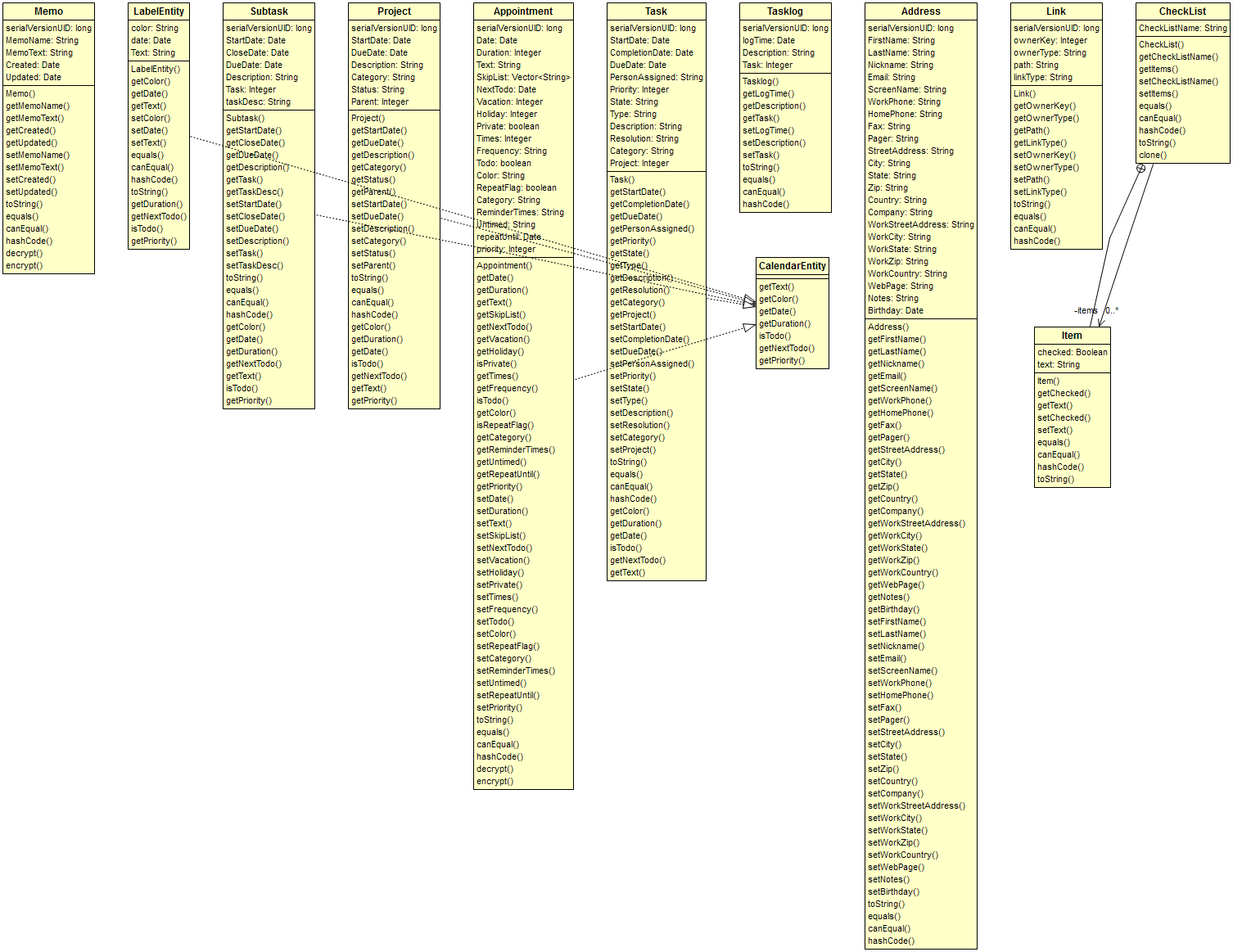
These extra classes are mainly not included in our design because we started from scratch and captured a simple basic set of entities based on Gall’s Law. This is a complex system that evolved for many years and these features are added in later versions.

There are 5 classes in our conceptual model that are not seen in the actual system model. These classes are:

* + AddressBook, ToDoList, ProjectList, MemoList: These classes wrap a collection object that holds a reference to the individual objects. Eg. Address, ToDo, Project and Memo. We intended to present collection objects in their respective entity classes as opposed to using arrays or vectors which is the solution for the actual system design.
  + Calender: This represents the calendar itself that is in association with calendar entity. In the actual system design this class is not present.
  1. Our Conceptual Class Diagram from M2



* 1. Actual System Class Diagram



# Code Smell and Possible Refactoring