Field Data Collection Final Report

Hee Su Chang

Name: Field Data Collection

Description: A service that allows scientists to easily and fluidly create their own templates to record data while out in the field for structured archival of data and metadata.

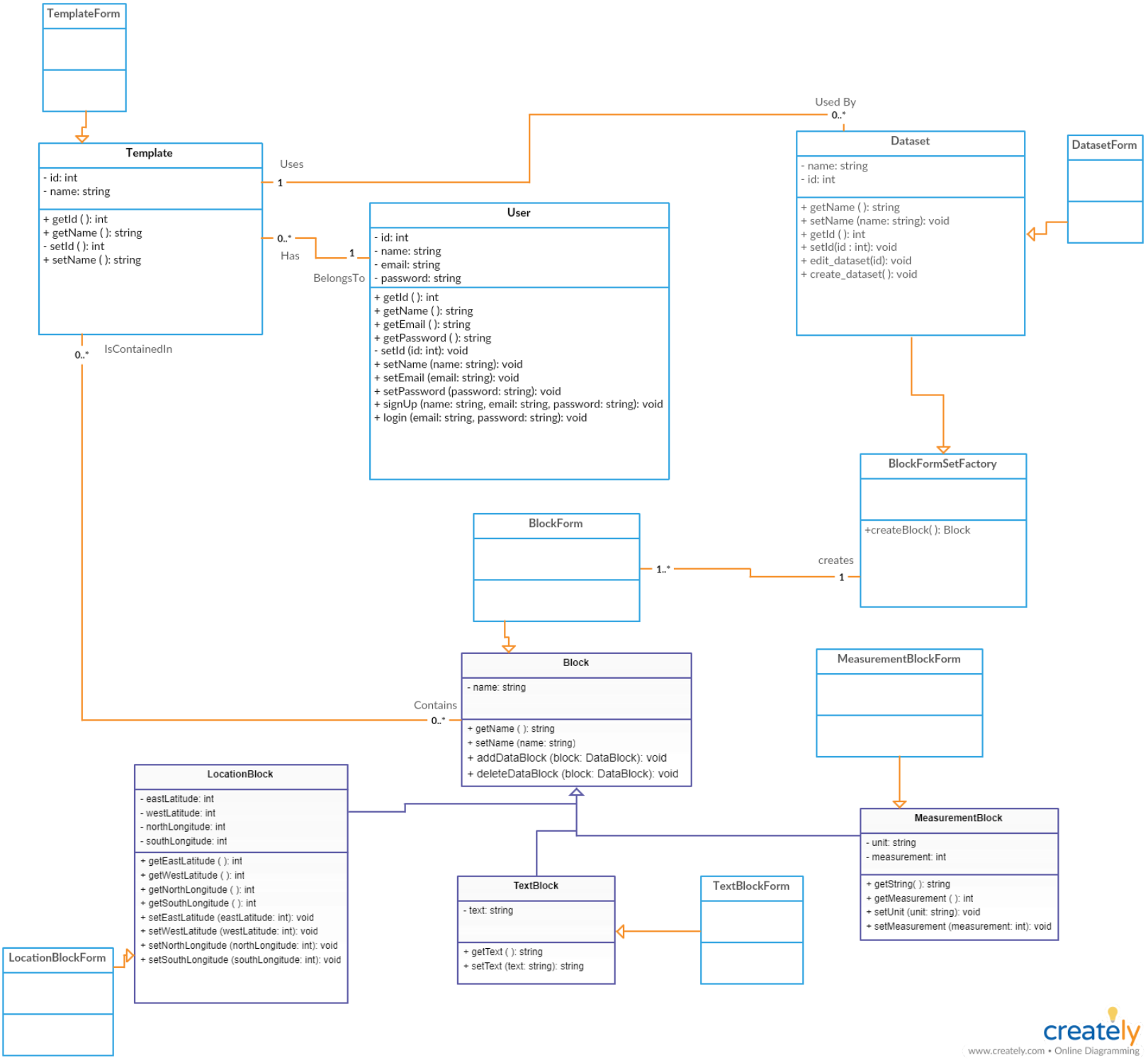
Implemented Features:

U01	User can create template with name unique to other templates
U02	User can add many various kinds of data blocks to template
U03	User can delete data blcoks added to template
U04	User can delete template
U06	User can create dataset using a template the user has made
U07	User can record data according to the chosen template in the dataset
U10	User can view dataset in website
U14	User can sign up with email, name, password
U15	User can log in with email, password

Unimplemented Features:

U05	User can create collection with name unique to other collections
U08	User can save dataset to collection
U09	User can delete dataset collection
U11	User can view dataset in JSON
U12	User can download dataset in JSON

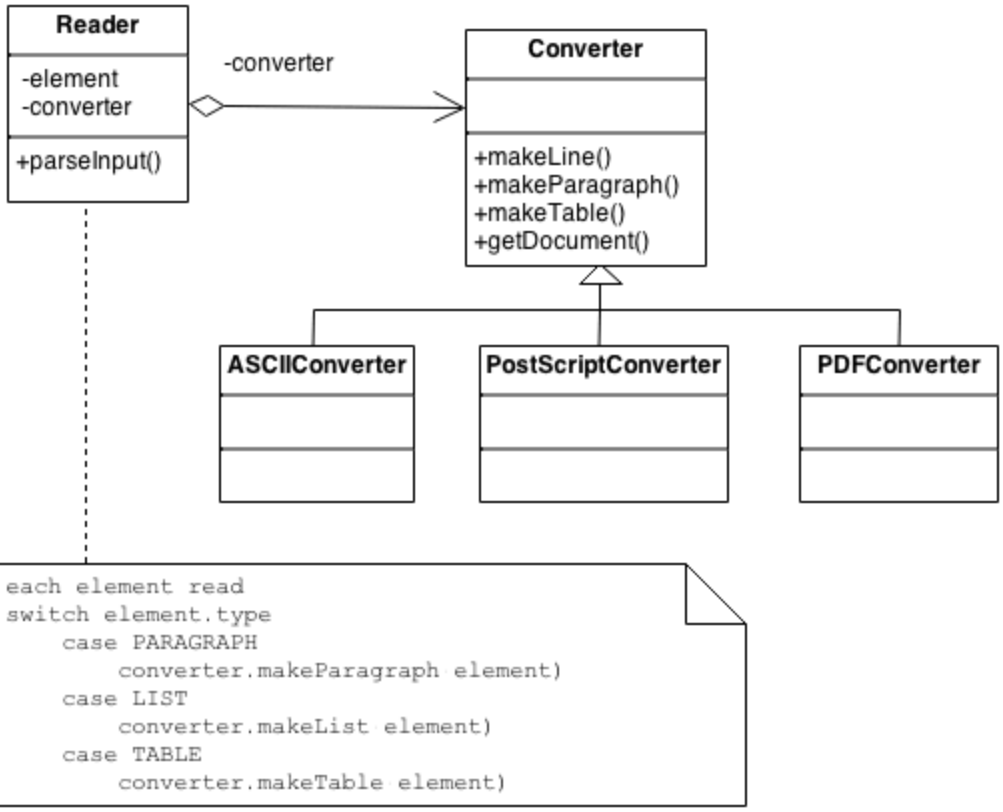
Final Class Diagram

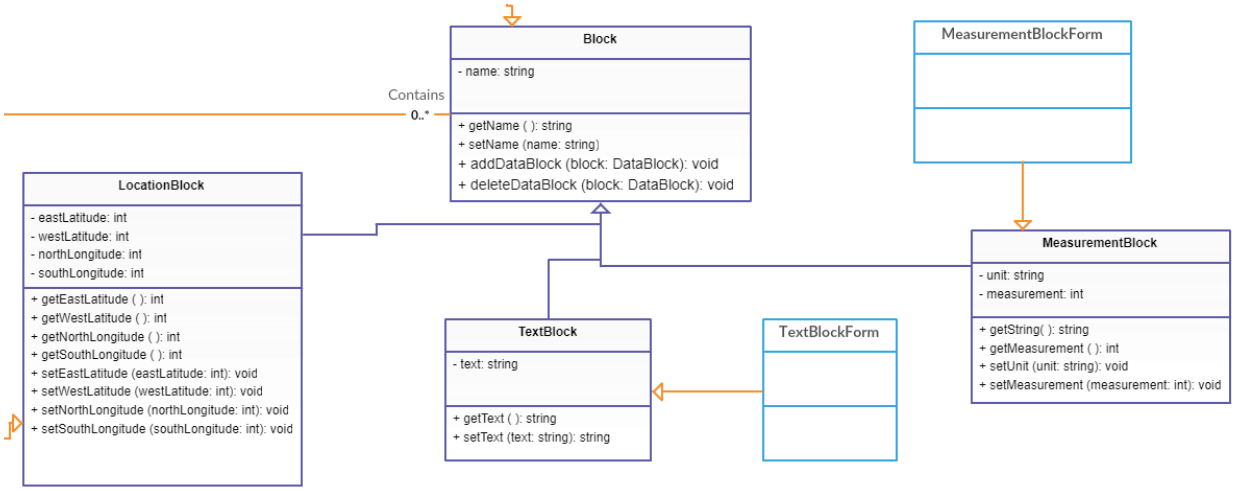


Quite a lot has changed in my final class diagram. I had to reduce a lot of the responsibilities in my User class because that didn't make sense. User is now only responsible for login/register functionality. I've also had to create a lot of smaller empty classes that inherit all the entities since Django requires these to utilize the form and formset features.

Implemented Design Patterns

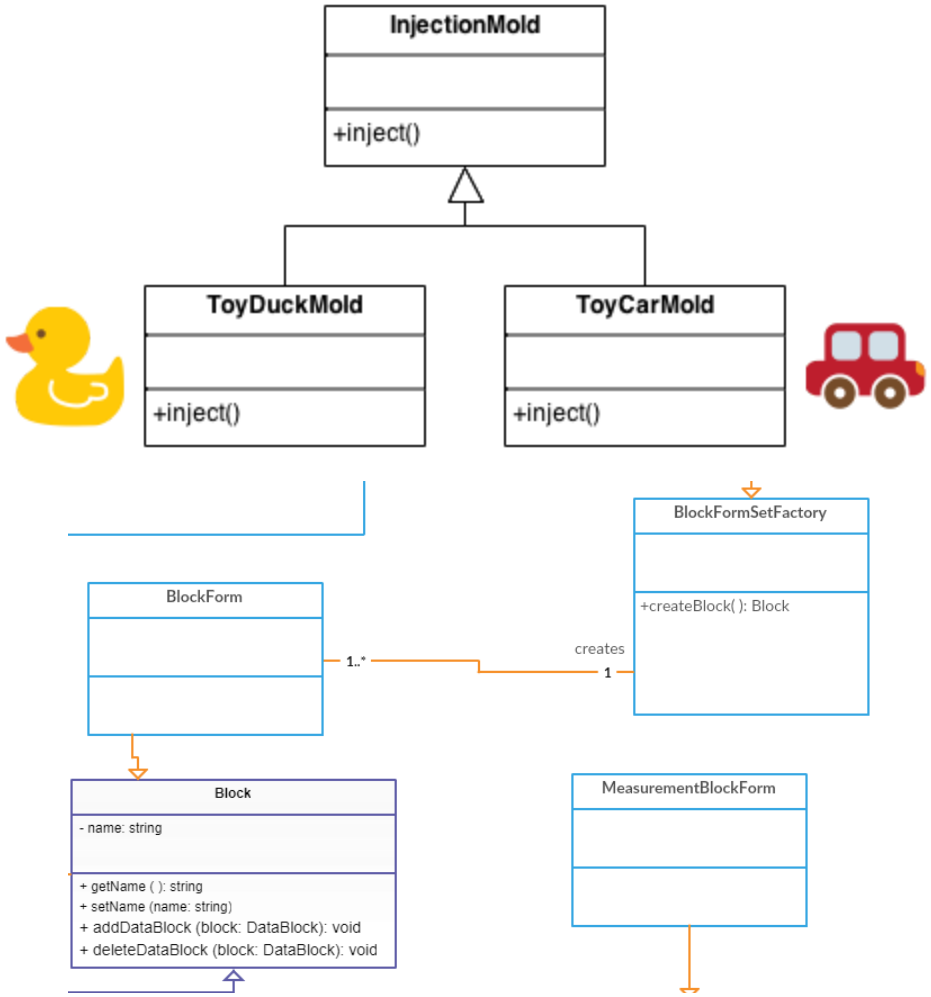
- Builder Design Pattern





I decided to use the Builder Design Pattern because I wanted template to be able to iterate through all the blocks regardless of the specific type of block. So, I implemented this using polymorphic models in Django to create a higher level representation (Block class) of all the different kinds of blocks that the Template class could reference.

- Factory Design Pattern



I decided to use the Factory Design Pattern to utilize the polymorphic formset factory in Django, so that I could display the forms for all the blocks in each dataset based on the template that the dataset is using. I utilized the polymorphic formset factory class to implement this, so I didn't actually create the class from scratch.

What I've Learned

I've learned that systems require a lot more classes than I expected. Not only do we need large entities, but there can be many smaller classes that represent the more definitive entities. I also learned more about how to delegate responsibilities to the correct entities as I saw this change from my first diagram and my final diagram in the User class. It would have been more useful to have better knowledge of overall web systems to design the system and I realized how much time and effort is required in this designing process which could have saved me much more time in the implementation.

Design Pattern Diagrams from https://sourcemaking.com/design_patterns