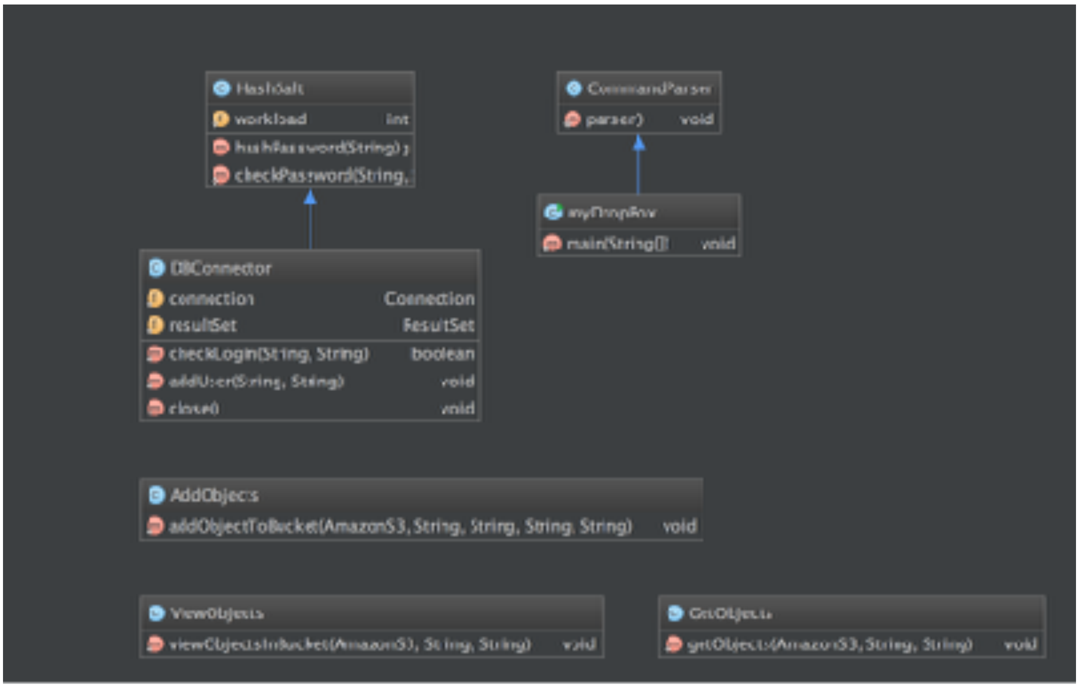
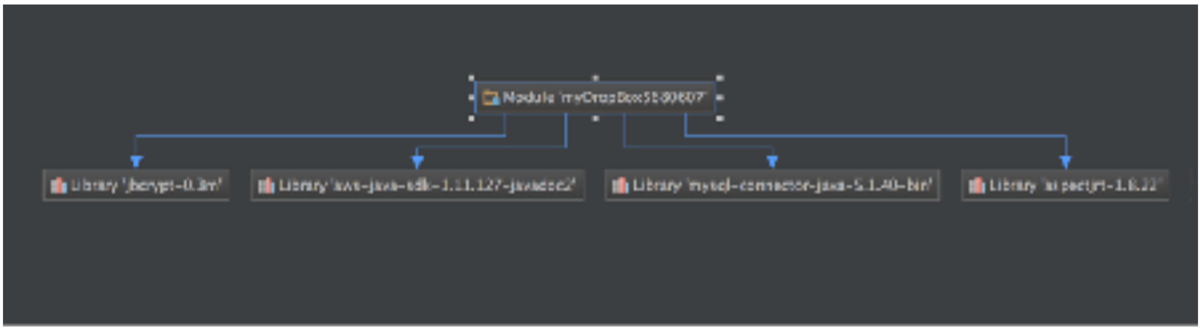
This command line based application consist of 9 classes.


HashSalt
workload int
hashPassword(String) ;
checkPassword(String,
CommandParser
parser() void
myDropBox
main(String[]) void
DBConnector
connection Connection
resultSet ResultSet
checkLogin(String, String) boolean
addUser(String, String) void
close() void
AddObjects
addObjectToBucket(AmazonS3, String, String, String, String) void
ViewObjects
viewObjectsInBucket(AmazonS3, String, String) void
GetObjects
getObjects(AmazonS3, String, String) void

I decided to separate addObjects, viewObjects and getObjects into different classes so that it becomes easier for me to debug. For DBConnector, I extended the HashSalt class so that hashPassword and checkPassword function can be used without having to declare a constructor. The myDropBox class extends CommandParser class so that parser() function can be used in the main method.

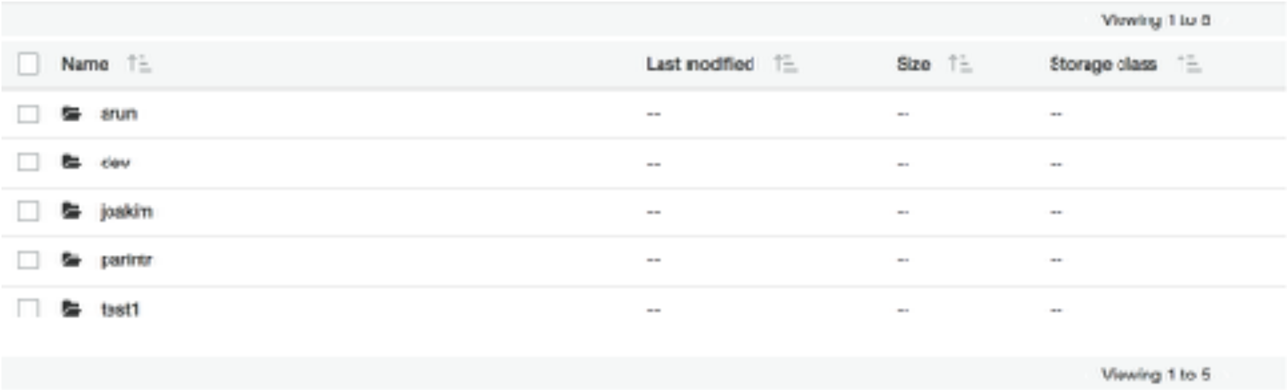
These are some of the libraries that I have used in my project.


Module 'myDropBox5880607'
Library 'jbcrypt-0.3m'
Library 'aws-java-sdk-1.11.127-javadoc2'
Library 'mysql-connector-java-5.1.40-bin'
Library 'aspectjrt-1.8.22'

At first, I tried to ‘mavenized’ my project. It seems to work just fine until I ran the jar file and got an exception. It was something to do with AWSCredentials, so I decided to switch back to the traditional way; adding the libraries to ./lib folder.

For database, I chose RDS that’s on AWS. The engine that is used for this project is MySQL.

Handling the upload, get and view cases. There are several ways to handle these cases, but the most effective way that I think would fit right for this application is to store the username is a variable and when the user uploads, get or view the objects; they’ll only see the objects that they’ve uploaded, etc. So how this work is, in the CommandParser class; parser() method; when user login, I store the username in the “user” variable. Now when the user uploads an object; they have to specify the key. Once they user specify the key, I add String username with a slash followed by the key. This would create something like a directory on S3.

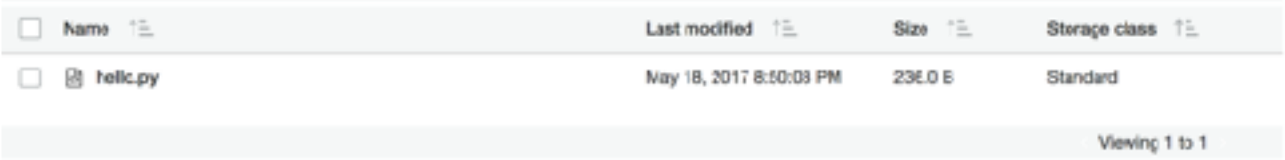


Viewing 1 to 5

□	Name	Last modified	Size	Storage class
□	arun	--	--	--
□	dev	--	--	--
□	joakim	--	--	--
□	parintr	--	--	--
□	test1	--	--	--

Viewing 1 to 5

So if I click ‘arun’, I’ll see the object inside it.



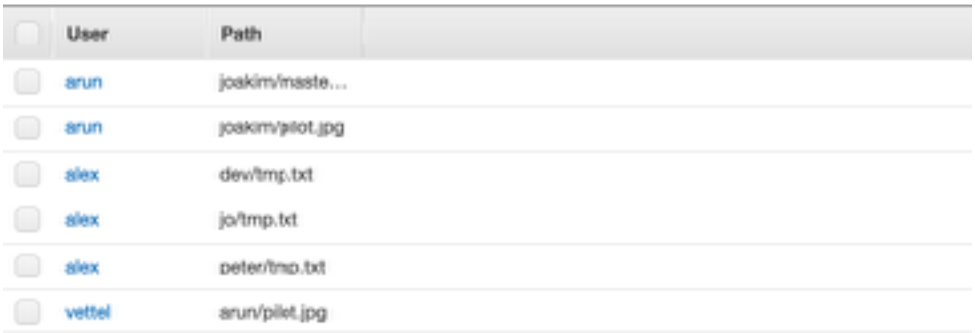
□	Name	Last modified	Size	Storage class
□	hello.py	May 18, 2017 8:50:03 PM	236.0 B	Standard

Viewing 1 to 1

This idea goes the same for viewing and downloading the object.

Share & ShareLogs class

ShareLogs is just the CRUD operations. The code in the Share class then uses the DynamoMapper to save an item and retrieve it.



□	User	Path
□	arun	joakim/maste…
□	arun	joakim/pilot.jpg
□	alex	dev/tmp.txt
□	alex	jo/tmp.txt
□	alex	peter/tmp.txt
□	vettel	arun/pilot.jpg

This is how the items are stored in the table. User is the hash key and path is the range key. When I retrieve the item, I check the user; whether it exist in the table or not. If it exist then add every path that is associated to that particular user to a list. So when a user want to view the files or download the shared files, I can just retrieve the path from this list.