AWS Startupkit

CIS 315

Professor: Stephan Bund

Charles M. Chong

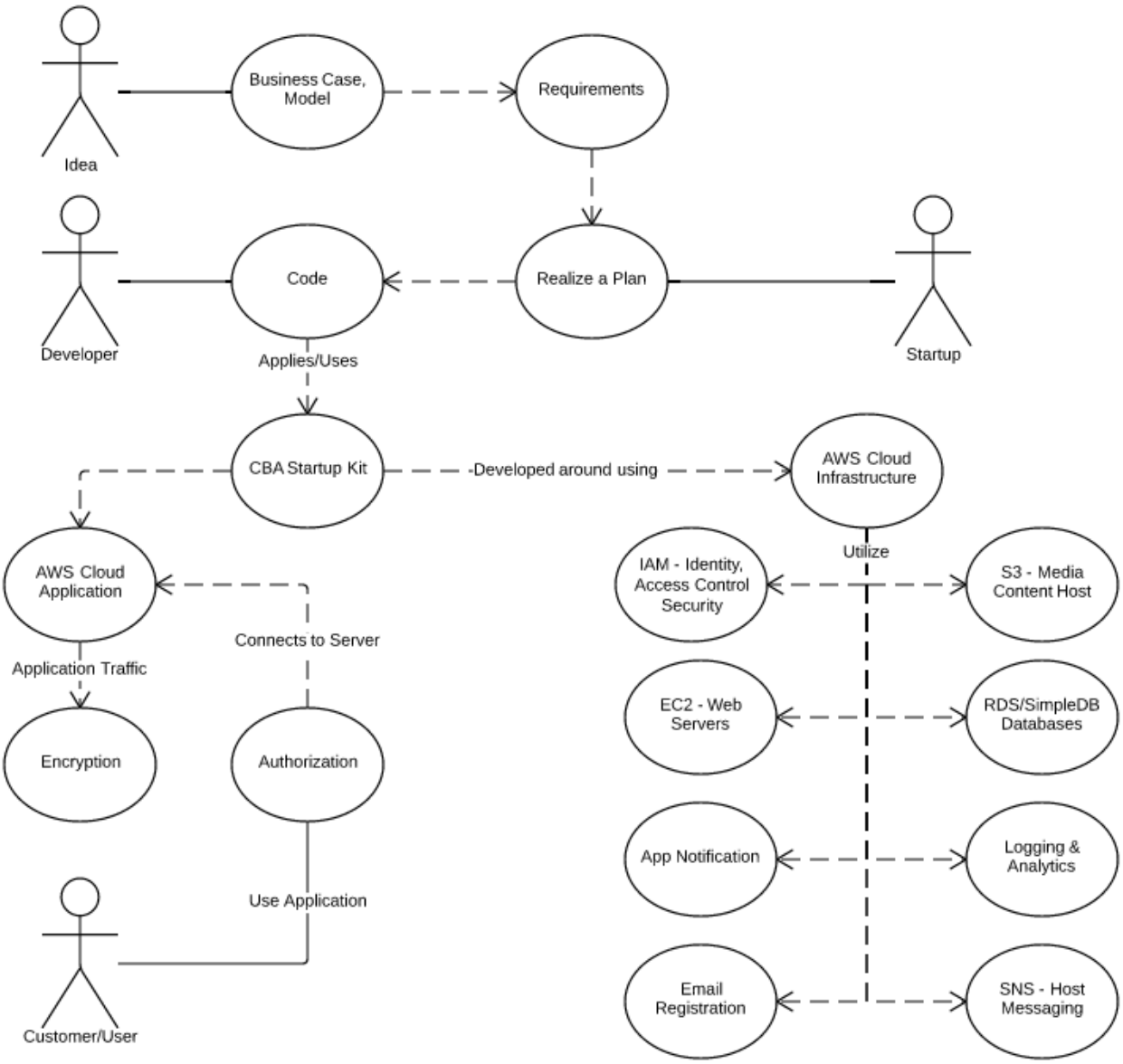
AWS Startup Kit Overview

Welcome to AWS Startup Kit!

AWS Startup Kit is a compiled API written in Java used to jumpstart developers to create and deploy an application with Amazon Web Services.

This kit is built around simple concepts of application development: running java against the AWS, database development, storage & content delivery, notifications, testing, and publishing.

The following diagram show the work flow and relationships between the elements of Amazon Web Servicces (AWS) with the CBA Startup Kit. These elements work together to create the process of developing a product involving a business startup.



Functional Requirements

1. Database

1.1. Connect to Amazon AWS SimpleDB Database

User able to run code on an AWS SimpleDB database server using AWS SimpleDB Client object. User can adjust the region they prefer to use.

1.2. Create a Domain in SimpleDB

User creates a Database known as a 'Domain' onto SimpleDB Server. Each Domain can hold attributes to define the critera of the stored values.

1.3. List all Domains of their Account

User runs a select statement against SimpleDB to list all 'Domains' that the user has access to.

1.4. Put data into Domain

User puts data into Domain such as values that fulfill the attribute properties.

1.5. Query data of a Domain

User runs a select statement to call data from an existing Domain. User can adjust the Query Expression to obtain their preferred data.

1.6. Delete values in attribute of a Domain

User can remove values from an attribute of a Domain containing the preferences made by the user.

1.7. Update an attribute of a Domain

User can update the contents of an attribute with the values made by the user.

1.8. Delete an attribute of a Domain

User can delete the contents of an attribute with the values made by the user.

1.9. Delete a Domain

User can delete a SimpleDB Domain and all of its containing contents.

2. Files

2.1. Push a file into Amazon AWS S3 Bucket

User can upload a file or folder contents into an AWS S3 Bucket.

2.2. Get a file from S3 Bucket

User can download a file from AWS S3 Bucket

2.3. Delete file from S3 Bucket

User can delete file from AWS S3 Bucket

Non-Functional Requirements

1. Operating System Requirements

1.1. Operate in Windows 7, Mac OS X 10.8+, Linux kernel 3.16 or later Environments

Most stable Operating Systems are preferred with at least 4 – 8 Gigabytes of RAM.

1.2. System is has latest OS and package Updates

Updates are important to patach all vulnerabilities, bugs, etc.

1.3. System able to create, modify, extract, and execute .java files

All these functions should be available by the recommended OS. Third-party apps are encouraged.

1.4. System will use Eclipse IDE for Java EE Developers with AWS Toolkit for Eclipse

This Toolkit is downloaded using Eclipse's 'Install New Software...' function.

1.5. System will use JAVA SE Development Kit 7u45 or later

Download directly from Oracle's Java SDK download webpage.

2. Performance Requirements

2.2. Response time to SimpleDB, S3, & Tests must be at least 7 seconds or less

Allows users to perform real-time responses against cloud server.

2.3. Data that is written to Database will be applied in Real-time

Clients would have positive outlook on an application that is responsive as soon as it is requested.

2.4. Maximum of 20 simultanous connections to limit bandwidth of AWS resources

While the application remains public, to lower the cost's of bandwidth use, limit simultananeous users.

2.5. Uptime of SimpleDB & S3 instances are at least 99% when application is in use

Users able to connect to the application with no issues.

3. Security Requirements

3.1. Login to Amazon AWS must have 2 Factor Authentication

AWS account secure from brute force attacks and prevents unauthorized users from using account.

3.2. AWS IAM default Root keys deleted

Removes the ability for a root account to have control of every services.

3.3. AWS IAM Profiles are created with limited access to Services

Limit access to of certain users to prevent unauthorized use.

3.4. Apply IAM password policy

Lengh minimum: 8, require an uppercase & lowercase letter, a number, & non-alphanumeric character.

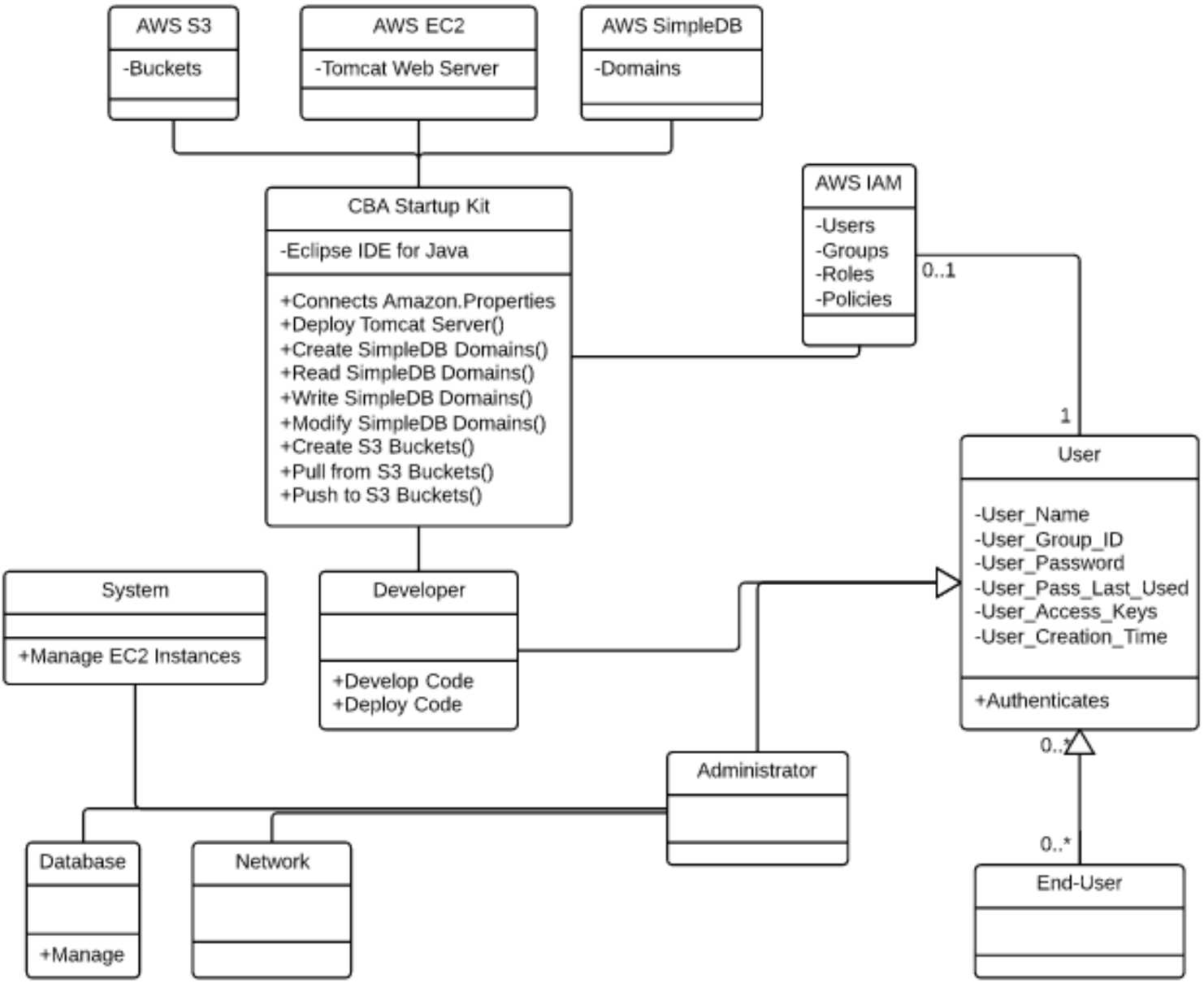
AWS StartupKit Packages (Update for Midterm)

Eclipse IDE for Java EE Developers

Required Packages List

```
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider;

import com.amazonaws.services.simpledb.AmazonSimpleDB;
import com.amazonaws.services.simpledb.AmazonSimpleDBClient;
import com.amazonaws.services.simpledb.model.Attribute;
import com.amazonaws.services.simpledb.model.BatchPutAttributesRequest;
import com.amazonaws.services.simpledb.model.CreateDomainRequest;
import com.amazonaws.services.simpledb.model.DeleteAttributesRequest;
import com.amazonaws.services.simpledb.model.Item;
import com.amazonaws.services.simpledb.model.PutAttributesRequest;
import com.amazonaws.services.simpledb.model.ReplaceableAttribute;
import com.amazonaws.services.simpledb.model.ReplaceableItem;
import com.amazonaws.services.simpledb.model.SelectRequest;
```

Basic Structural Model



Static Architecture

Server Overview

1. AWS SimpleDB – Database

Using a SQL database to hold user and application data. Used for authentication and storage of key values that represent user data.

2. AWS S3 – Storage

Holds application and media user data. Assigns keys to all media so that it can be identified by pull and push commands from client and server.

3. AWS EC2 – Web Server

Computing Instance hosted by AWS running Tomcat Servlet. Makes application publically accessible by any device over the internet.

Java and AWS code (Updated for Midterm)

database.java

```
public class database {

	private static AmazonSimpleDB sdb = new AmazonSimpleDBClient (new
ClasspathPropertiesFileCredentialsProvider());
	private static Region usWest2 = Region.getRegion(Regions.US_WEST_2);

	/**
	 * Prints out Attributes for selected Item
	 * @param itemIdentifier the request that matches the selected expression
	 */

	public static void getAttributesForItem(String domainName)
	{
		sdb.setRegion(usWest2);

		String selectExpression = "select * from `" + domainName + "` where Category
= 'Clothes'";
		SelectRequest selectRequest = new SelectRequest(selectExpression);
		for(Item item : sdb.select(selectRequest).getItems()){
			System.out.println("  Item");
			System.out.println("   Name: " + item.getName());
			for(Attribute attribute : item.getAttributes())
			{
				System.out.println("    Attribute");  //SDB stores and returns
key/value pairs
				System.out.println("    Name: " + attribute.getName()); //JSON
				System.out.println("    Value: " + attribute.getValue());
			}
		}
	}
	/**
	 * Item Sample data with attributes to add in a Domain
	 * @return boolean confirms method works
	 */
	@SuppressWarnings("unused")
	private static List<ReplaceableItem> createSampleData()
	{
		List<ReplaceableItem> sampleData = new ArrayList<ReplaceableItem>();

		sampleData.add(new ReplaceableItem("Item_01").withAttributes(
				new ReplaceableAttribute("Category", "Clothes", true),
				new ReplaceableAttribute("Subcategory", "Sweater", true),
				new ReplaceableAttribute("Name", "Cat Hair Sweater", true),
				new ReplaceableAttribute("Color", "Siamese", true),
				new ReplaceableAttribute("Size", "Small", true),
				new ReplaceableAttribute("Size", "Medium", true),
				new ReplaceableAttribute("Size", "Large", true)
				));

		sampleData.add(new ReplaceableItem("Item_02").withAttributes(
				new ReplaceableAttribute("Category", "Clothes", true),
```

```
                    new ReplaceableAttribute("Subcategory", "Pants", true),
                    new ReplaceableAttribute("Name", "Designer Jeans", true),
                    new ReplaceableAttribute("Color", "Paisley Acid Wash", true),
                    new ReplaceableAttribute("Size", "30x32", true),
                    new ReplaceableAttribute("Size", "32x34", true),
                    new ReplaceableAttribute("Size", "48x40", true)
                    ));

        sampleData.add(new ReplaceableItem("Item_03").withAttributes(
                    new ReplaceableAttribute("Category", "Clothes", true),
                    new ReplaceableAttribute("Subcategory", "Pants", true),
                    new ReplaceableAttribute("Name", "Sweatpants", true),
                    new ReplaceableAttribute("Color", "Pink", true),
                    new ReplaceableAttribute("Color", "Red", true),
                    new ReplaceableAttribute("Color", "Chartreuse", true),
                    new ReplaceableAttribute("Size", "Small", true),
                    new ReplaceableAttribute("Year", "2006", true),
                    new ReplaceableAttribute("Year", "2007", true)
                    ));

        return sampleData;
}

/**
 * Item New data with attributes to add in a Domain
 * @return boolean confirms method works
 */
private static List<ReplaceableItem> createData(String newItemName)
{
        List<ReplaceableItem> newData = new ArrayList<ReplaceableItem>();
        newData.add(new ReplaceableItem(newItemName).withAttributes(
                    new ReplaceableAttribute("Category", "Tools", true),
                    new ReplaceableAttribute("Subcategory", "Auto", true),
                    new ReplaceableAttribute("Name", "Wrench", true),
                    new ReplaceableAttribute("Color", "Silver", true),
                    new ReplaceableAttribute("Color", "Blue", true),
                    new ReplaceableAttribute("Color", "Red", true),
                    new ReplaceableAttribute("Size", "Large", true),
                    new ReplaceableAttribute("Year", "2006", true),
                    new ReplaceableAttribute("Year", "2007", true)
                    ));
        return newData;
}
/**
 * Creates a Domain in Amazon SimpleDB
 * Makes domain a variable to use for other methods
 * @param domainName the name of the domain created by method
 * @return boolean confirms method works
 */
public static boolean createDB(String domainName)
{
              //CREATE A DOMAIN TO USE, TO PUSH DATA
              //String myDomain = "Hump"; //a sample domain, for test purposes
              sdb.createDomain(new CreateDomainRequest(domainName));
              return true;
        }

/**
```

```
        * Executes query to Database and returns results
        * @param domainName the name of the domain created by method
        * @param query the statement that will be executed to the database
        * @return boolean confirms method works
        */
       public static boolean read(String domainName,String query)
       {
              sdb.setRegion(usWest2);
              String selectExpression = query;
              SelectRequest selectRequest = new SelectRequest(selectExpression);
              for (Item item : sdb.select(selectRequest).getItems())
              {
                     System.out.println("  Item");
                     System.out.println("   Name: " + item.getName());
                     for( Attribute attribute : item.getAttributes())
                     {
                            System.out.println("      Attribute"); //SDB JSON ARRIVES IN
NAME/VALUE PAIRS
                            System.out.println("       Name:  " + attribute.getName());
                            System.out.println("        Value: " + attribute.getValue());
                     }
              }
              return true;
       }

       /**
        * Writes data into selected Domain of an item
        * @param domainName the database table
        * @param itemName the name of the item
        * @param itemAttribWrite written attribute to item
        * @return boolean confirms method works
        */
       public static boolean write(String domainName, String itemName, String
itemAttribWrite)
       {
              sdb.setRegion(usWest2);
              sdb.batchPutAttributes(new BatchPutAttributesRequest(domainName,
createData(itemName)));
              return true;
       }
       /**
        * Updates attribute information of a Item in the selected Domain
        * @param domainName the name of the domain created by method
        * @param itemName the name of the item
        * @param itemAttribUpdate updated attribute properties
        * @return boolean confirms method works
        */
       public static boolean update(String domainName, String itemName, String
itemAttribUpdate)
       {
              sdb.setRegion(usWest2);
              List<ReplaceableAttribute> replaceableAttributes = new
ArrayList<ReplaceableAttribute>();
              replaceableAttributes.add(new ReplaceableAttribute("Size", itemAttribUpdate,
true));
              sdb.putAttributes(new PutAttributesRequest(domainName, itemName,
replaceableAttributes));
              return true;
```

```
	}

	/**
	 * Deletes attribute information of a Item in the selected Domain
	 * @param domainName the name of the domain created by method
	 * @param itemName the name of the item
	 * @return boolean confirms method works
	 */
	public static boolean delete(String domainName, String itemName)
	{
		sdb.setRegion(usWest2);
		sdb.deleteAttributes(new DeleteAttributesRequest(domainName, itemName));
		return true;
	}

}//end class
```

file.java

```
public class file {
	private static Region usWest2 = Region.getRegion(Regions.US_WEST_2);
	private static AWSCredentials credentials = new
ProfileCredentialsProvider("charlesc").getCredentials();
	private static AmazonS3 s3 = new AmazonS3Client(credentials);
	public file() {
	}
	/**
	 * Pushes selected file to the AWS S3 bucket
	 * @param bucketName name of S3 bucket container
	 * @param key the directory containing media content
	 * @param fileLocation the directory containing file on local system
	 * @throws IOException checks for errors, throws exception to continue if error
found
	 * @return boolean confirms method works
	 */
	public static boolean uploadFile(String bucketName, String key, String
fileLocation)throws IOException {
		try {
		s3.setRegion(usWest2);
		File file = new File(fileLocation);
		s3.putObject(new PutObjectRequest(bucketName, key, file));

	}  catch (AmazonServiceException ase) {
	  System.out.println("Caught an AmazonServiceException, which means your request
made it "
			+ "to Amazon S3, but was rejected with an error response for some
reason.");
	  System.out.println("Error Message:    " + ase.getMessage());
	  System.out.println("HTTP Status Code: " + ase.getStatusCode());
	  System.out.println("AWS Error Code:   " + ase.getErrorCode());
	  System.out.println("Error Type:       " + ase.getErrorType());
	  System.out.println("Request ID:       " + ase.getRequestId());
    } catch (AmazonClientException ace) {
	  System.out.println("Caught an AmazonClientException, which means the client
encountered "
			+ "a serious internal problem while trying to communicate with S3, "
			+ "such as not being able to access the network.");
	  System.out.println("Error Message: " + ace.getMessage());
    }
		return true;
	}

	/**
	 * Downloads file from S3 Bucket
	 * @param bucketName name of S3 bucket container
	 * @param key the directory containing media content
	 * @return boolean confirms method works
	 * @throws IOException checks for errors, throws exception to continue if error
found
	 */
	public static boolean downloadFile(String bucketName, String key) throws IOException
{
		s3.setRegion(usWest2);
		S3Object object = s3.getObject(new GetObjectRequest(bucketName, key));
```

```
			displayTextInputStream(object.getObjectContent());
			return true;
		}
		/**
		 * Delete file in S3 Bucket
		 * @param bucketName name of S3 bucket container
		 * @param key the directory containing media content
		 * @return deletes file in selected S3 Bucket
		 */
		public static boolean deleteFile(String bucketName, String key) {
			s3.setRegion(usWest2);
			s3.deleteObject(bucketName, key);
			return true;
		}

		/**
		 * @param input displays output of read buffer
		 * @throws IOException checks for errors, throws exception to continue if error
found
		 */
		private static void displayTextInputStream(InputStream input)throws IOException {
			BufferedReader reader = new BufferedReader(new InputStreamReader(input));
			while (true) {
				String line = reader.readLine();
				if (line == null)
					break;
				System.out.println("    " + line);
			}
			System.out.println();
		}

}
```

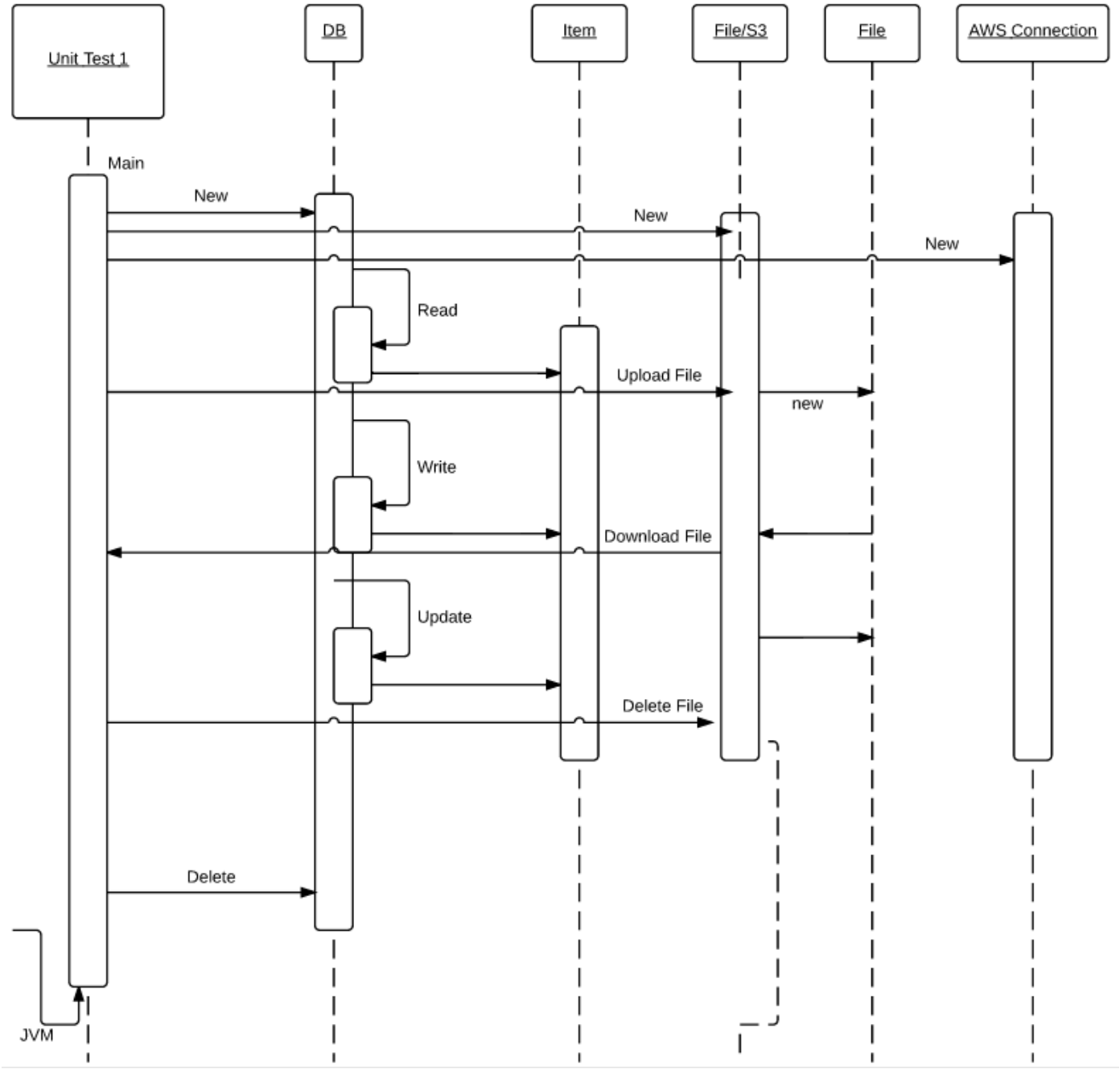
Unit_Test_1.java (Midterm)

```
public class Unit_Test_1 {

	private static AmazonSimpleDB sdb = new AmazonSimpleDBClient (new
ClasspathPropertiesFileCredentialsProvider());
	private static database db = new database();//putting into place the objects I wish
to test
	@SuppressWarnings("static-access")
	public static void main(String[] args) throws IOException
	{
		file f = new file(); //bucketName, key, directory path of file
		if(f.uploadFile("bucket-test-now","MyObjectKeyNew",
"./src/startupkit/TestFileName.txt") == true)
		{
			System.out.println("File Upload OK \n");
			if(f.downloadFile("bucket-test-now", "MyObjectKeyNew") == true)
			{
				System.out.println("File Download OK \n");
				if((f.deleteFile("bucket-test-now", "MyObjectKeyNew")) == true)
				{
					System.out.println("Delete Download OK \n");
				}
			}
		}
		else //To signal that a failure took place
		{
			System.out.println("Looks like the file op failed");
		}
		//Creating DB & Listing Current DB Test
		if(db.createDB("TestSimpleDB") == true) //boolean result
		{
			System.out.println("Database creation OK \n");
			for (String domainName : sdb.listDomains().getDomainNames()) //FETCH A
LIST OF ALL MY DATABASE DOMAINS
			{
				System.out.println("--" + domainName);
			}
			//Reading Test
			if(db.read("MyStore", "select * from MyStore") == true)
			{
				System.out.println("Database Item Read OK \n");
				if(db.write("MyStore", "ItemNameTest", "MyStore") == true){
					System.out.println("Database write OK \n");
					System.out.println("Showing Written Item");
	System.out.println("-----------------------------------------");
					db.read("MyStore", "select * from MyStore");
					if (db.update("MyStore","ItemNameTest","Small") == true){
						if (db.delete("MyStore", "ItemNameTest") == true);
						System.out.println("Database Item Update OK \n");
	System.out.println("-----------------------------------------");
						db.read("MyStore", "select * from MyStore");
						System.out.println("Database Item Delete OK \n");
					} } } } } }
```

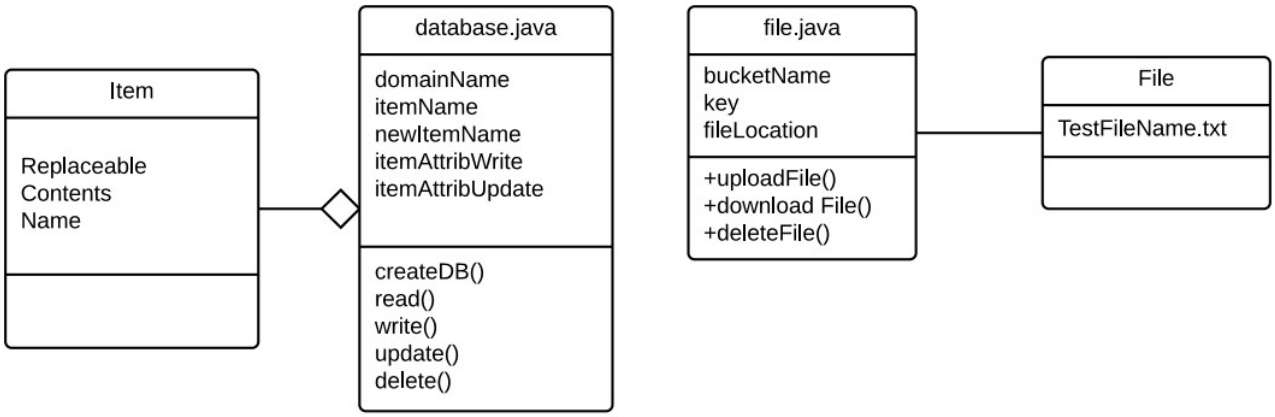
Unit_Test_1 Console Output (Midterm)

```
File Upload OK

    this is a test file in root directory of project package

File Download OK

Delete Download OK

Database creation OK

--MyStore
--TestSimpleDB
Database Item Read OK

Database write OK

Showing Written Item
-----------------------------------------
  Item
   Name: ItemNameTest
      Attribute
       Name:  Name
        Value: Wrench
      Attribute
       Name:  Category
        Value: Tools
      Attribute
       Name:  Subcategory
        Value: Auto
      Attribute
       Name:  Color
        Value: Red
      Attribute
       Name:  Color
        Value: Blue
      Attribute
       Name:  Color
        Value: Silver
      Attribute
       Name:  Size
        Value: Large
      Attribute
       Name:  Year
        Value: 2007
      Attribute
       Name:  Year
        Value: 2006
Database Item Update OK

-----------------------------------------
  Item
   Name: ItemNameTest
      Attribute
       Name:  Name
        Value: Wrench
      Attribute
       Name:  Category
        Value: Tools
      Attribute
       Name:  Subcategory
        Value: Auto
      Attribute
       Name:  Size
        Value: Small
      Attribute
       Name:  Color
        Value: Blue
      Attribute
       Name:  Color
        Value: Red
      Attribute
       Name:  Color
        Value: Silver
      Attribute
       Name:  Year
        Value: 2006
      Attribute
       Name:  Year
        Value: 2007
Database Item Delete OK
```

Sequence Diagram (Midterm)



Class Diagram (Midterm)



Javadocs (Updated)