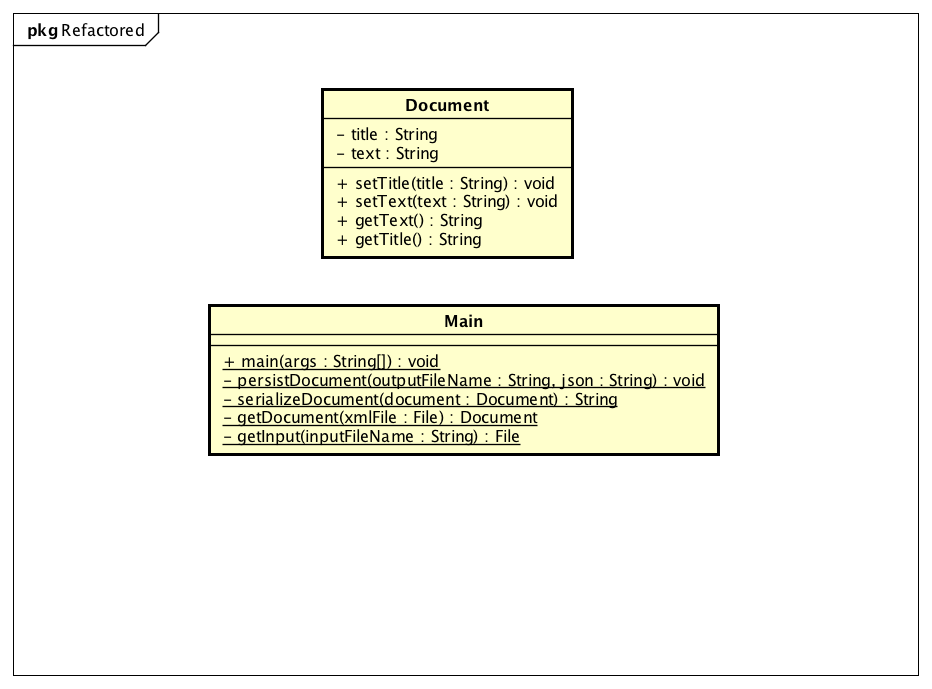
SOLID PRINCIPLES

1.- CREAMOS UN PAQUETE QUE SE LLAME REFACTORED.

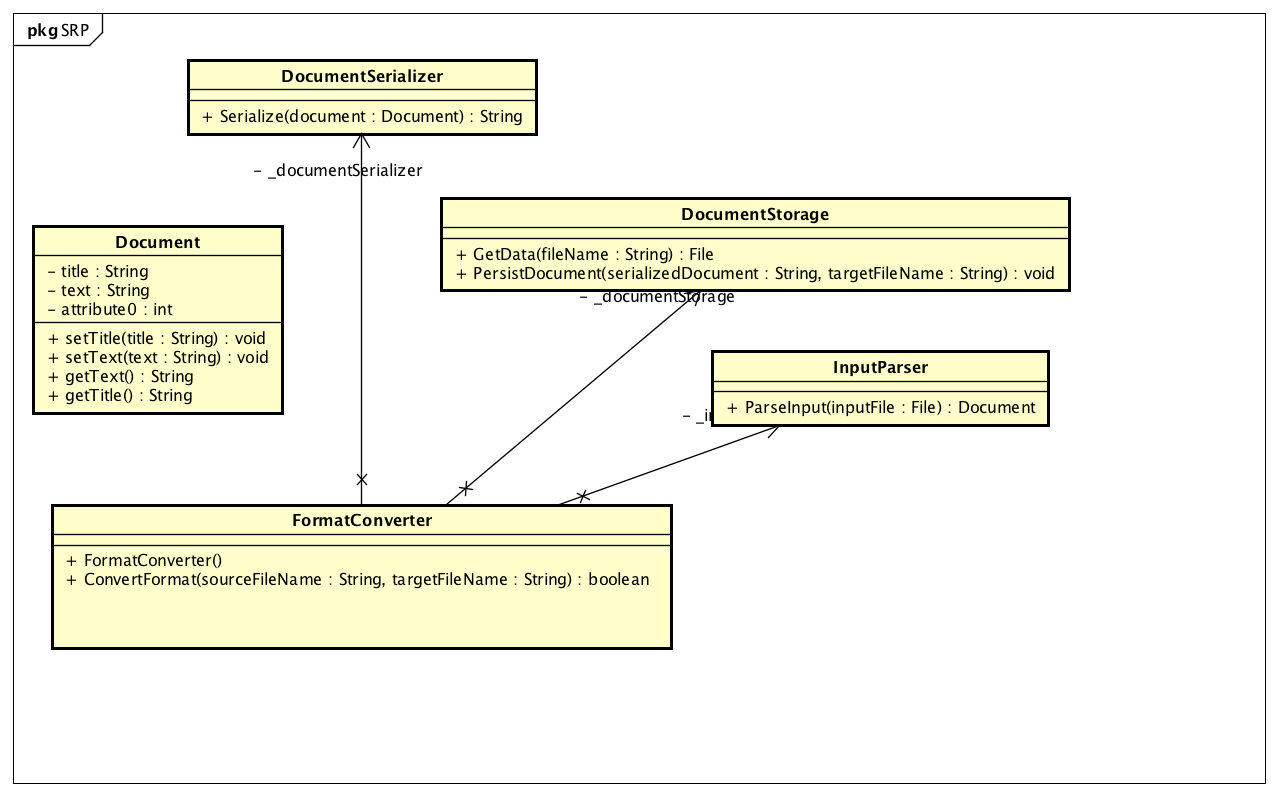
2.- COPIAMOS AMBAS CLASES DOCUMENT Y MAIN

3.- EXTRAEMOS 4 METODOS EN EL FICHERO MAIN COMO VEMOS EN EL UML

SINGLE RESPONSIBILITY PRINCIPLE

1.- CREAMOS UN PAQUETE QUE SE LLAME SRP.

2.- COPIAMOS AMBAS CLASES DOCUMENT Y MAIN DEL PROYECTO ANTERIOR

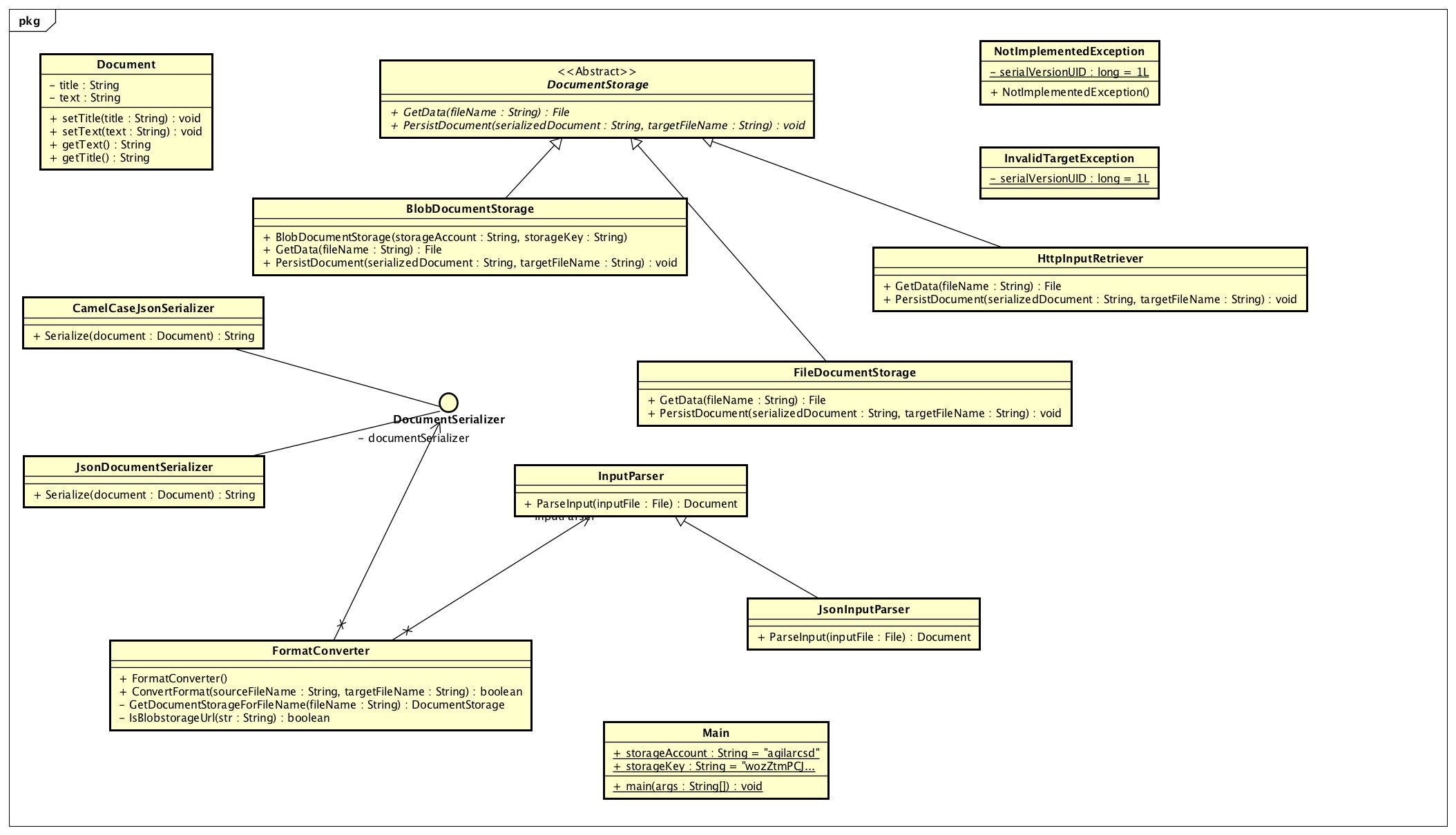
3.- IMPLEMENTAR EL SIGUIENTE UML.

OPEN CLOSE PRINCIPLE

1.- CREAMOS UN PAQUETE QUE SE LLAME OCP.

2.- COPIAMOS LAS CLASES DEL PROYECTO ANTERIOR EN EL NUEVO PAQUETE

3.- IMPLEMENTAR EL SIGUIENTE UML.



public class NotImplementedException extends RuntimeException{

private static final long serialVersionUID = 1L;

public NotImplementedException(){}

}

public class InvalidTargetException extends RuntimeException {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

}

**CamelJsonSerializer**

public String Serialize(Document document) {

Gson gson = new GsonBuilder().setFieldNamingPolicy(FieldNamingPolicy.UPPER\_CAMEL\_CASE).create();

return gson.toJson(document);

}

**BlobDocumentStorage**

CloudBlobClient blobClient;

public BlobDocumentStorage(String storageAccount, String storageKey)

throws InvalidKeyException, URISyntaxException {

String storageConnectionString = String.format("DefaultEndpointsProtocol=http;AccountName=%s;AccountKey=%s",

storageAccount, storageKey);

CloudStorageAccount account = CloudStorageAccount.parse(storageConnectionString);

blobClient = account.createCloudBlobClient();

}

@Override

public File GetData(String fileName) {

if (!fileName.startsWith(blobClient.getStorageUri().toString())) {

throw new InvalidTargetException();

}

URI uri = null;

try {

uri = new URI(fileName);

} catch (URISyntaxException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return new File(uri);

}

@Override

public void PersistDocument(String serializedDocument, String targetFileName) {

try {

URI uri = new URI(targetFileName);

String containerName = uri.getPath().substring(1, uri.getPath().indexOf("/", 1) - 1);

CloudBlobContainer container = blobClient.getContainerReference(containerName);

container.createIfNotExists();

BlobContainerPermissions containerPermissions = new BlobContainerPermissions();

// Include public access in the permissions object.

containerPermissions.setPublicAccess(BlobContainerPublicAccessType.CONTAINER);

// Set the permissions on the container.

container.uploadPermissions(containerPermissions);

// Create or overwrite the "myimage.jpg" blob with contents from a

// local file.

CloudBlockBlob blob = container.getBlockBlobReference(

targetFileName.replace(blobClient.getStorageUri() + containerName + "/", ""));

blob.uploadText(serializedDocument);

} catch (Exception e) {

// Output the stack trace.

e.printStackTrace();

}

}

**FormatConverter**

InputParser inputParser = null;

JsonDocumentSerializer documentSerializer = null;

public FormatConverter() {

inputParser = new JsonInputParser();

documentSerializer = new JsonDocumentSerializer();

}

public boolean ConvertFormat(String sourceFileName, String targetFileName)

throws IOException, ParserConfigurationException, SAXException, InvalidKeyException, URISyntaxException {

boolean converted = false;

File input = null;

DocumentStorage sourceDocumentStorage = GetDocumentStorageForFileName(sourceFileName);

input = sourceDocumentStorage.GetData(sourceFileName);

Document doc = inputParser.ParserInput(input);

String serializedDoc = documentSerializer.Serialize(doc);

DocumentStorage targetDocumentStorage = GetDocumentStorageForFileName(targetFileName);

targetDocumentStorage.PersistDocument(serializedDoc, targetFileName);

converted = true;

return converted;

}

private DocumentStorage GetDocumentStorageForFileName(String fileName)

throws InvalidKeyException, URISyntaxException {

if (IsBlobstorageUrl(fileName))

return new BlobDocumentStorage(Main.storageAccount, Main.storageKey);

if (fileName.startsWith("http"))

return new HttpInputRetriever();

return new FileDocumentStorage();

}

private boolean IsBlobstorageUrl(String str) {

return str.startsWith(String.format("https://%s.blob.core.windows.net/", Main.storageAccount));

}

**Main**

public static final String storageAccount ="csdagilar";

public static final String storageKey ="e1niPCjt/i3bU/wllm4Bfqe9nP4JpAiCGEqAu+RqITwxF8K5jeZlIT7SGzZ+ah8WP3qfkRFeI8ivSjibOJffuA==";

public static void main(String[] args) throws IOException, ParserConfigurationException, SAXException, InvalidKeyException, URISyntaxException {

String current = new File( "." ).getCanonicalPath();

String sourceFileName = current + "\\Output Documents\\Document1.json";

String targetFileName = "https://csdagilar.blob.core.windows.net/csdagilar/Document<NombreUsuario>.json";

FormatConverter formatConverter = new FormatConverter();

if (!formatConverter.ConvertFormat(sourceFileName, targetFileName)) {

System.out.println("Conversion failed...");

}

else

System.out.println("Conversion done...");

}

Probar en Azureexplorer

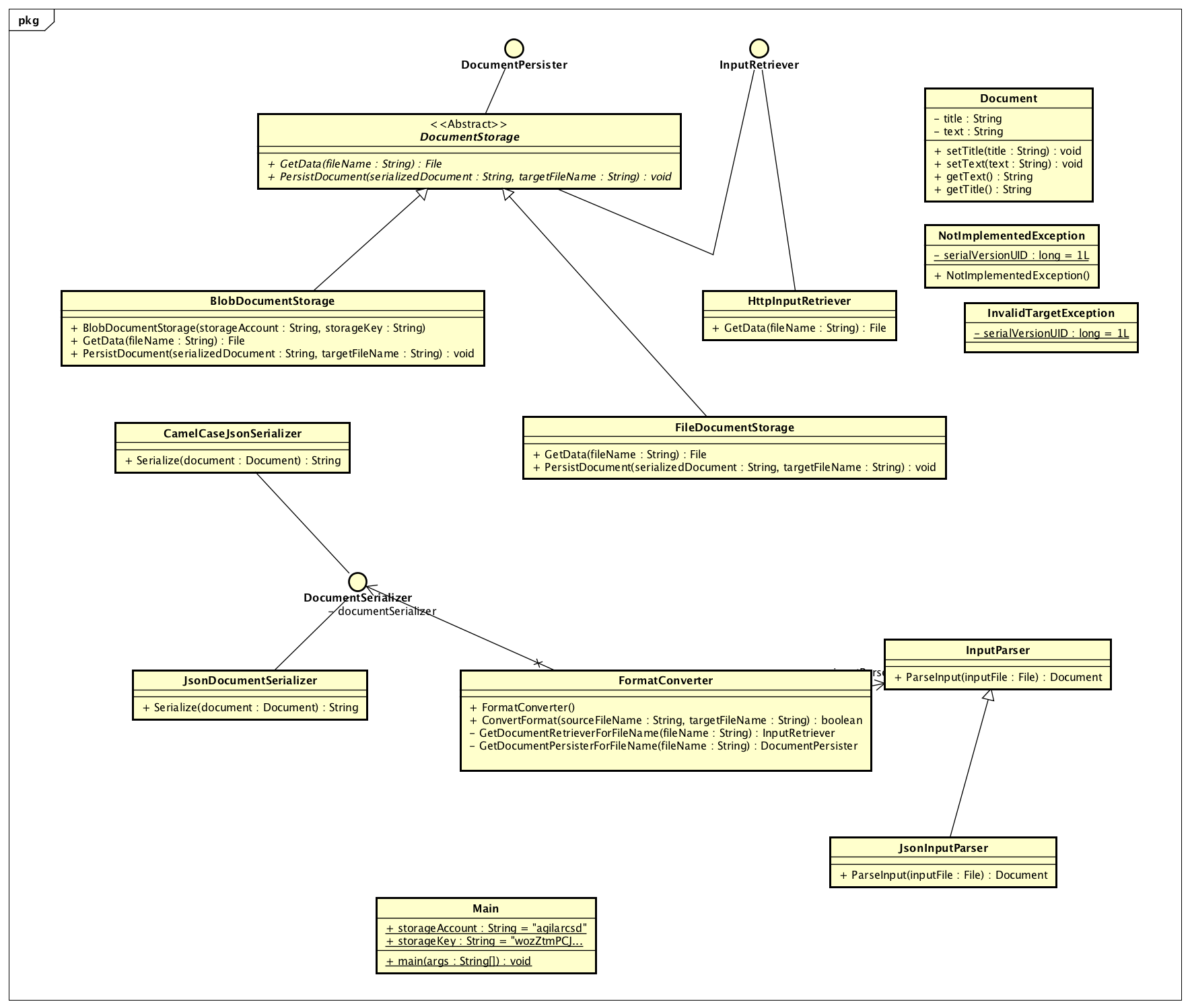
<http://azurestorage.azurewebsites.net/>

INTERFACE SEGREGATION PRINCIPLE

1.- CREAMOS UN PAQUETE QUE SE LLAME ICP.

2.- COPIAMOS LAS CLASES DEL PROYECTO ANTERIOR EN EL NUEVO PAQUETE

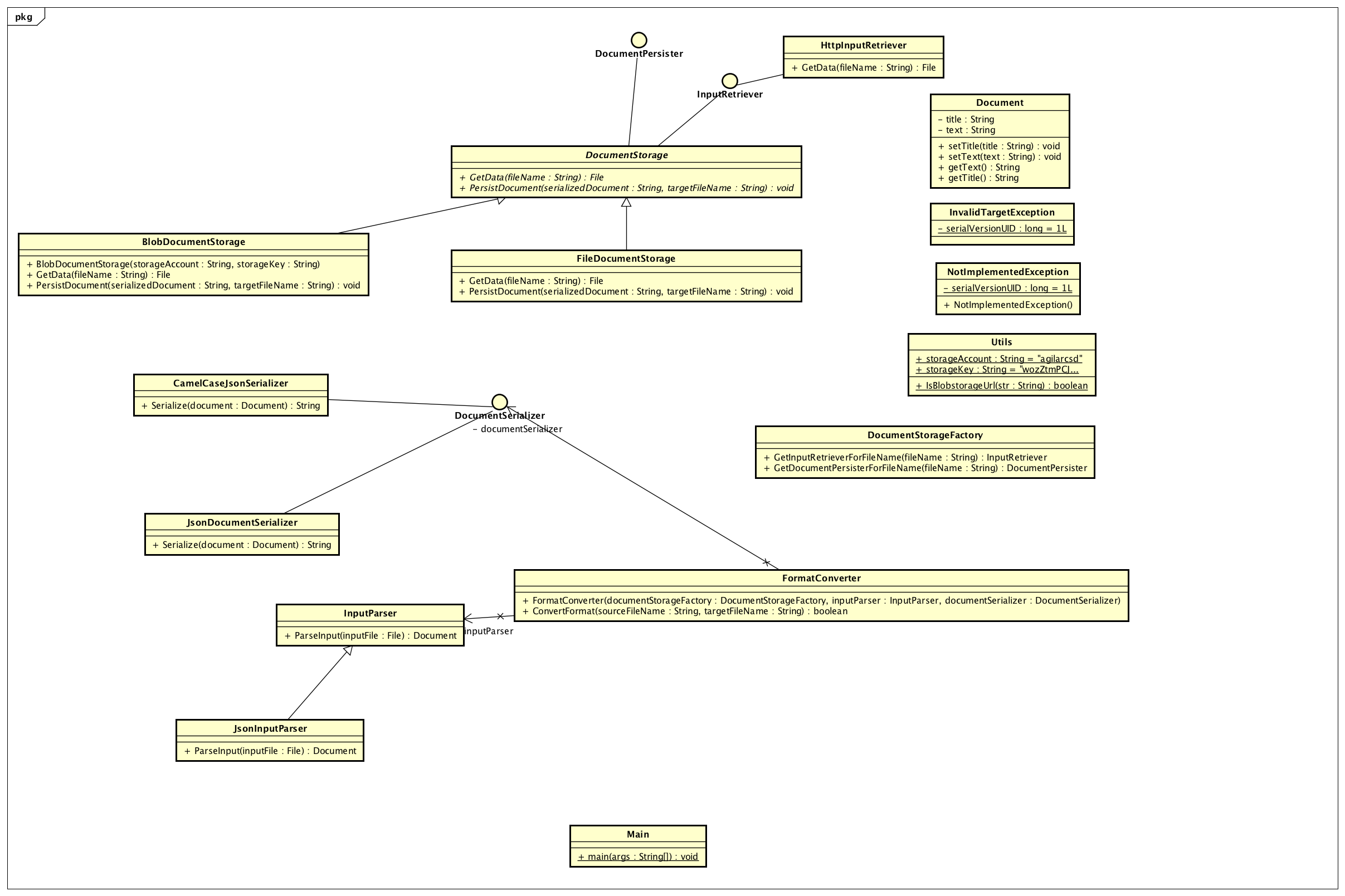
3.- IMPLEMENTAR EL SIGUIENTE UML.



INVERSION DE DEPENDENCIAS PRINCIPLE (FACTORY MANUAL)

1.- CREAMOS UN PAQUETE QUE SE LLAME DIP.

2.- COPIAMOS LAS CLASES DEL PROYECTO ANTERIOR EN EL NUEVO PAQUETE

3.- VAMOS A CREAR LA FACTORIA MANO DE INVERSION DE DEPDENDENCIAS PARA ELLO CREAMOS UN PAQUETE QUE SE LLAME MANUAL.

INVERSION DE DEPENDENCIAS PRINCIPLE (Guice)

1.- CREAMOS UN PAQUETE QUE SE LLAME DIP.IoC

2.- COPIAMOS LAS CLASES DEL PROYECTO ANTERIOR EN EL NUEVO PAQUETE

3.- LO HACEMOS TODOS JUNTOS

4.- CREAMOS UNA NUEVA CLASE Module

public class FormatConverterModule extends AbstractModule {

@Override

protected void configure() {

// TODO Auto-generated method stub

bind(InputParser.class).to(JsonInputParser.class);

bind(DocumentSerializer.class).to(CamelCaseJsonSerializer.class);

bind(StorageFactory.class).to(DocumentStorageFactory.class);

}

}

@Inject

public FormatConverter

@Singleton

public class CamelCaseJsonSerializer

Main

Injector injector = Guice.createInjector(new FormatConverterModule());

FormatConverter formatConverter = injector.getInstance(FormatConverter.class);