Separating out Files

1. Till Now we are doing everything in a single file whether it is a main function , other function or global variables . All of these are declared in one single file.

2. Our code is broadly composed of two things i.e. a.) Algorithm and b.) Data.

3. Example : There is an Algorithm to sort an array and there is Data Like Hashmap , 1-D array or 2-D array and a lot many other things.

4. Let there are 2 Algorithms (A1 and A2) and 3 Data (D1 ,D2 and D3). it is not necessary that all the algorithm uses all the data, example A1 uses D1 and D2 while A2 uses D3. Hence we can separate out A1 with D1 and D2 , similarly A2 with D3 .

Not All the Data and algorithms are interconnected to each other.

5. We can divide or separate multiple functions into different classes which can be further divided into modules and packages.

6. Example : Create Renaissance with following functionality

→ login (Username + Password)

→ Fetch (Free Trial Modules , Subscribed User Modules)

There are two ways to implement the above functionality i.e. in a single file or in separate files.

We will stick to separate files , we can make 3 files for above functionality which are

UserCredentialsValidator : This Module contains the required data for authentication of the user and it does not need to know anything about Modules for Free trial User or Subscribed User.

ModulesRetreiver : This class file contains List of free Trial Modules and Subscribe User Modules. Functions to getModules on the basis of username and their subscription.

Renaissance : This class interacts with its dependencies i.e. UserCredentialsValidator and ModulesRetreiver. It just creates their objects and with that it gets the job done by calling those objects accordingly.

These above 3 classes are in different files , but these will be placed in a single folder or package.

7. Benefits of this are that it enhances the maintainability to the next level . Example if we know that there is a bug in login part then we just only have to debug 1 file i.e. UserCredentialsValidator . But in a single file where everything is written in a single piece of code then you have to go through complete code.

8. Different Packages can be assigned to different developers for testing and maintenance.

9.Testing becomes much easier.

A Basic Intro to Class

A Basic Intro to class

1. The basic construct that helps us separate our code into different parts is class.

2. The basic purpose of the class is to put the related data and algorithms (methods / Functions) together.

3. Classes consist of data and methods which are related to each other and contained in an entity.

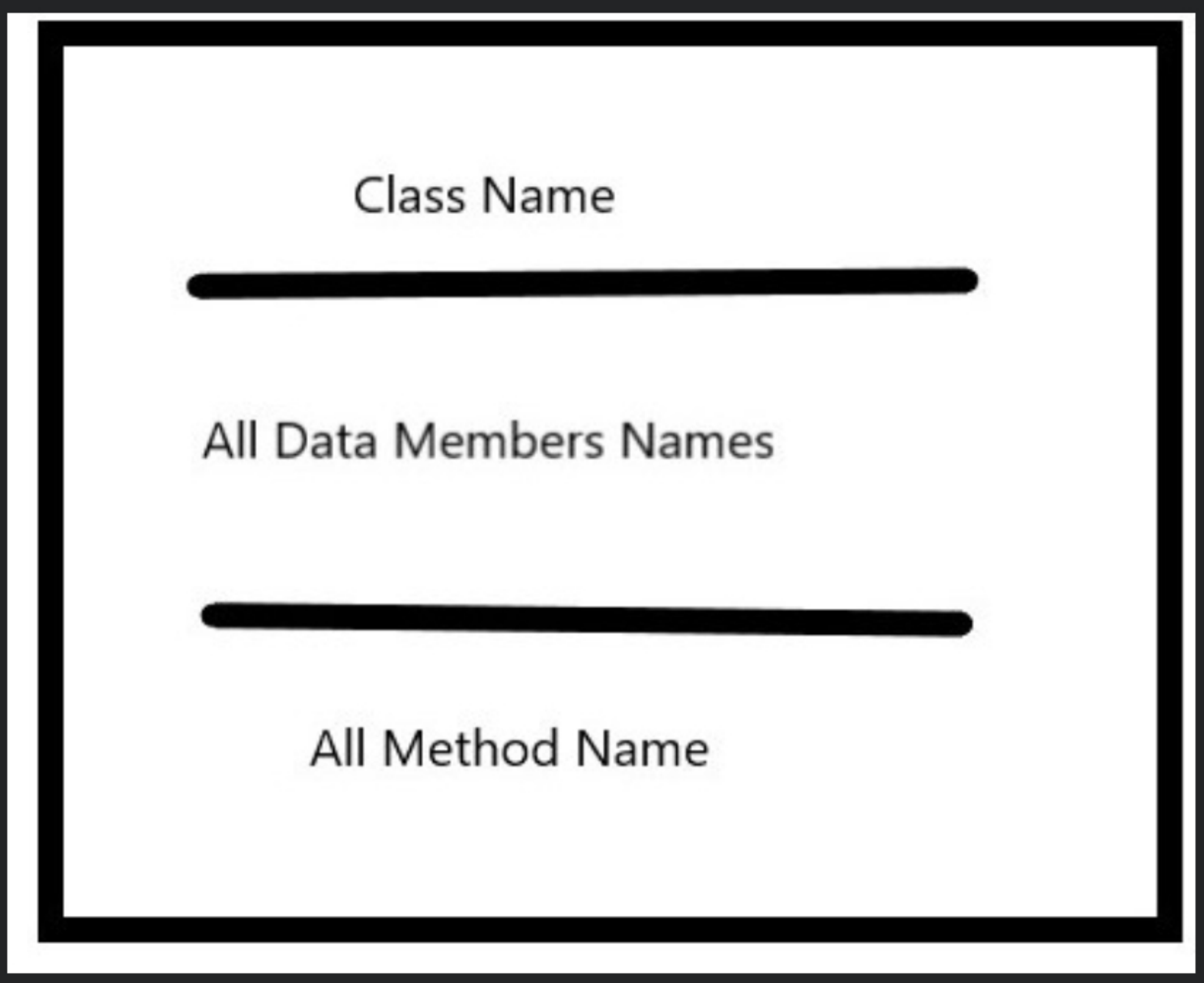
4. In a software world if we want to design a system then there are multiple entities that it will have , and each entity represents a class and each of this class have their own data and own sets of functions.

5. Example : e-commerce websites like amazon have different entities which are payment Gateways , orders , shipments , cart etc . because they all deserve different classes.

6. File system is a software which is composed of multiple entities.

7. We can invoke the methods of the class using the objects of the class.

8. In further lectures, when we will be learning about class diagrams , we will have to draw the logical representation of class which is shown below



9. The dependencies between these multiple classes / entities is shown by an Arrow in a class diagram.

10. The Art behind a good LLD is to be able to think about the correct entities and the right kind of relationship / dependencies between them.

Static Methods

1. A static method means it can be called without creating an instance of the class.

2. One should not create instances of a class which only contains static methods. Constructors of such classes should be made private.

3. A non static variable/method can't be accessed in static methods.

4. A static variable/method can be accessed in non static methods.

5. Static variables/methods/classes are mostly used to perform procedural functions.

6. Static variables are initialised during the class loading step.