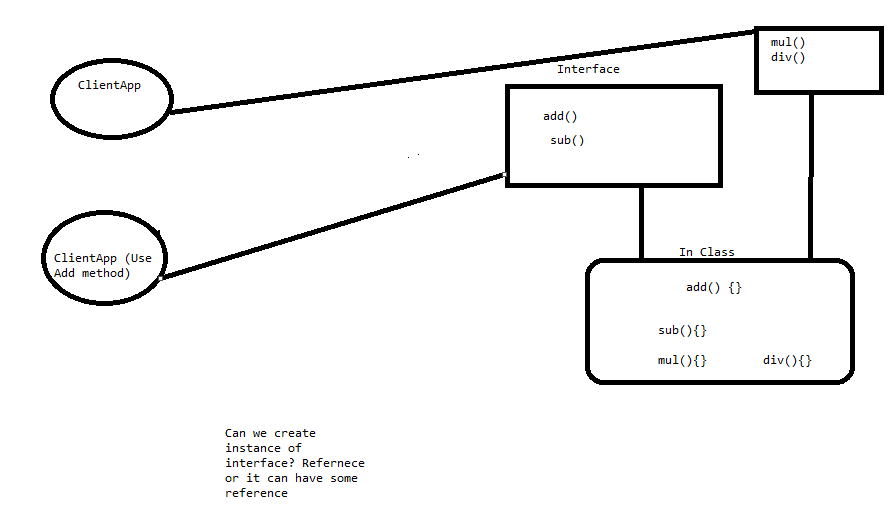
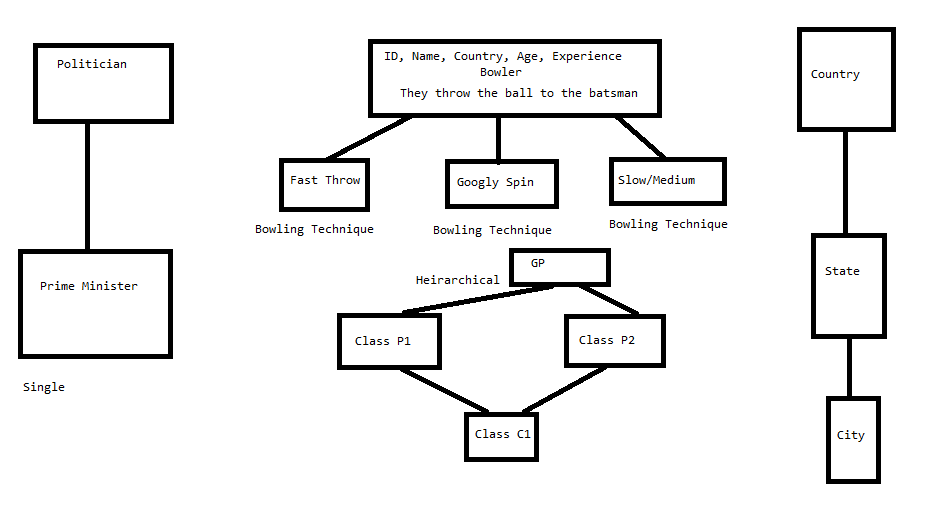
1. OOP
   1. Features
      1. Encapsulation
      2. Abstraction
      3. Inheritance
      4. Polymorphism
   2. Main reason behind OOP is
      1. Increase flexibility
      2. Increase the maintainability
      3. Security
      4. Increase Productivity
      5. High Quality
      6. Decrease development time

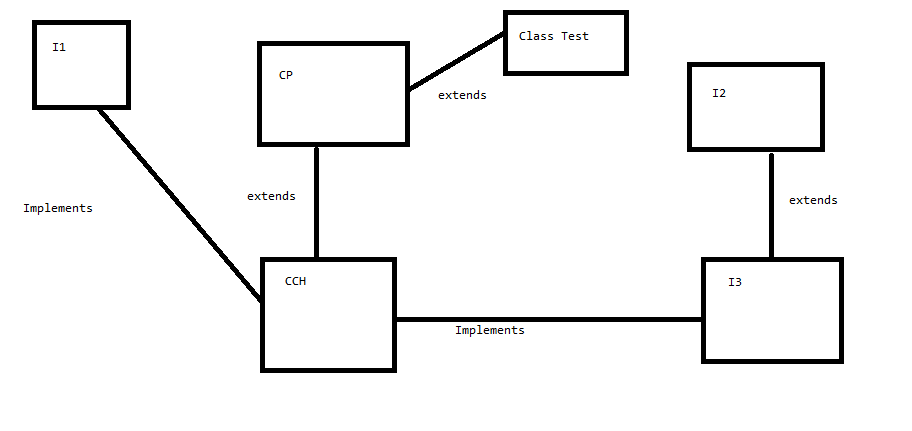
* 1. We represent every logic through the object by relating it with real environment for which you are doing the automation.

1. How the OOP features can be implemented inside the program?
   1. We can implement all the features with the help of
      1. Class Concepts
      2. Access Specifiers



1. Inheritance
   1. Generalization
      1. All the general implementations to be put together
   2. Specialization
      1. All the Specific implementation to be distributed across specific object
   3. We are achieving Reusability
   4. Types of inheritance
      1. Single
      2. Multi-level
      3. Hierarchical





Create a class 'Animal' with following properties

    - id

    - name

    - Height

    - Weight

    - numberofLegs

    - Define the property methods for all the properties

    - Define the method for accepting and displaying the Animal details

Create another class 'Cat' which will have all the properties related to Cat

    - This class will inherit the Animal class

    - This class will overide the base class methods for accepting and displaying

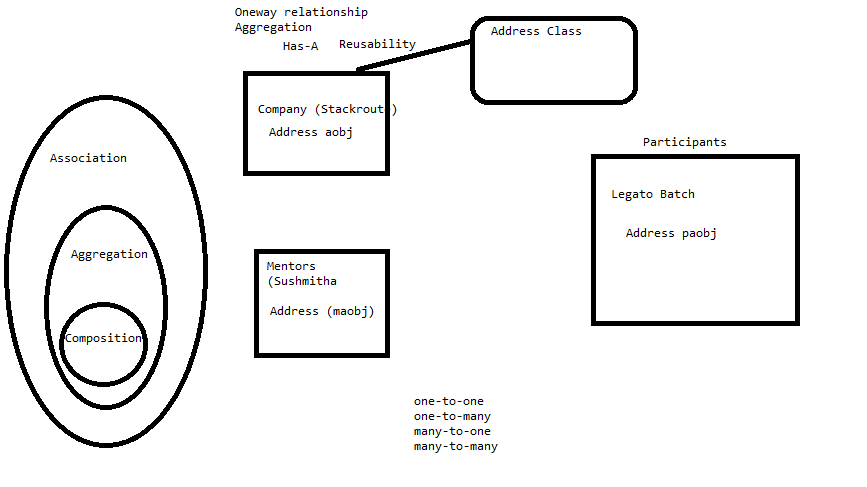
    - Use toString method to display the properties

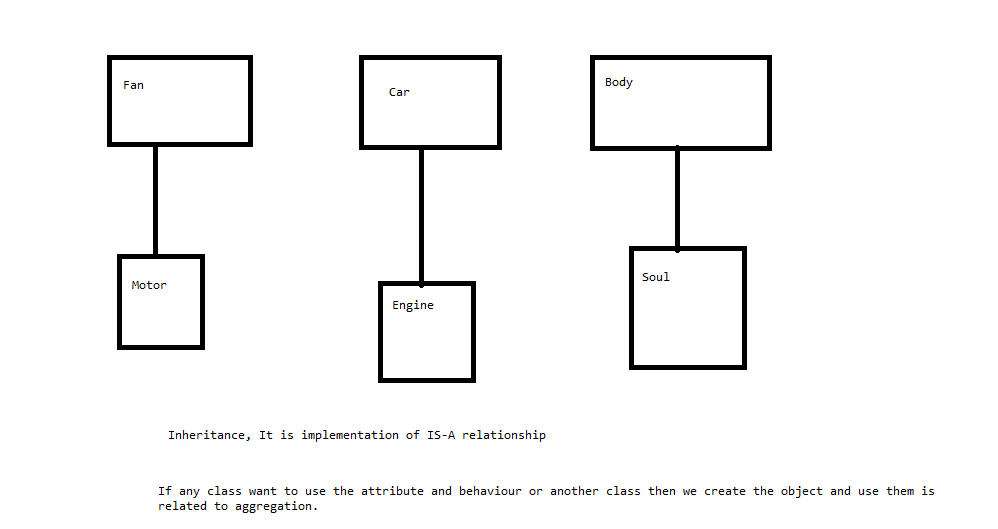
Create two class name client

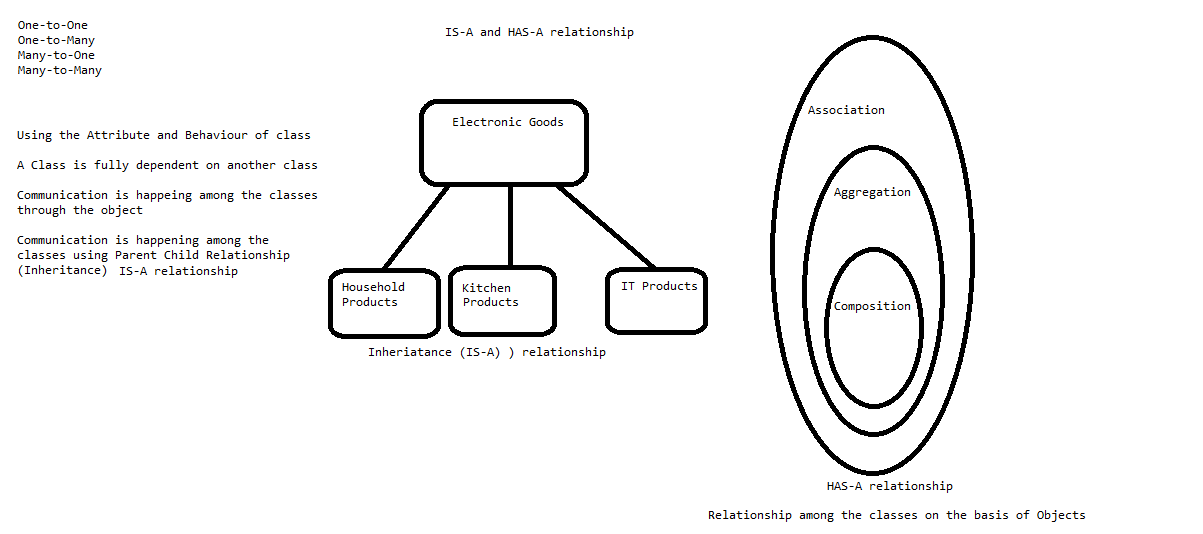
    - ClientA can accept data

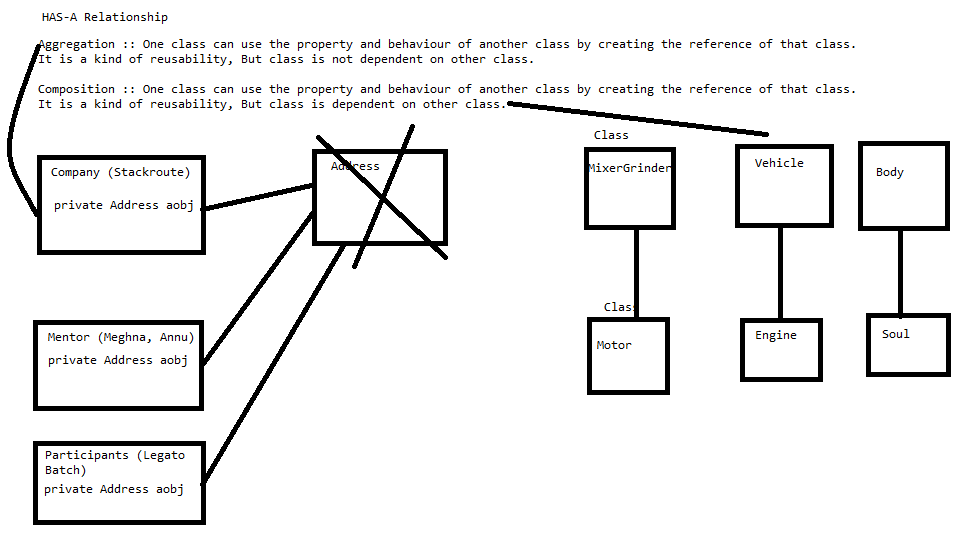
    - ClientB can display data

Use 100% abstraction

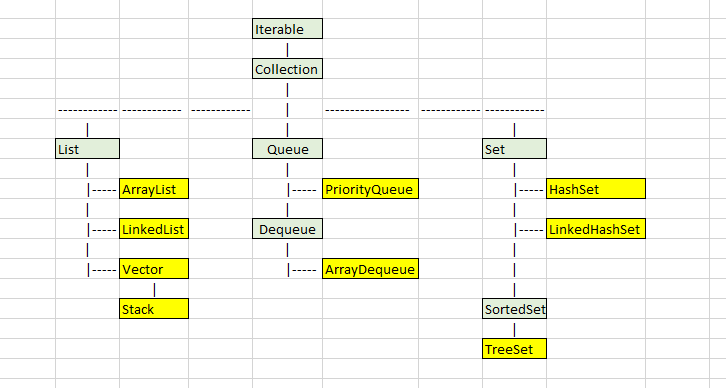


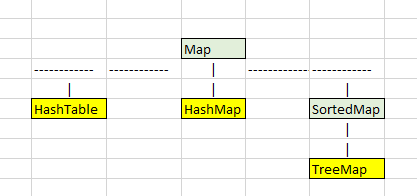






1. Data Structures and Java Collections
   1. How the data is getting stored inside the memory?
      1. Variable
      2. Array
      3. Link List
   2. Processing the Data
      1. Stack 🡪 LIFO
      2. Queue 🡪 FIFO
      3. Tree 🡪 Hierarchy of Data (pre-order, post-order and in-order)
   3. We already understood the complexity involved in managing the Data as well as processing the data.
2. Java introduced the concept related to Data Structures by providing Java Collections Framework.
   1. What is Framework?
      1. It provides ready support to implement any concept by using standard approach. It provides library which contain predefined classes and Interfaces to support the implementation of your concept. All the functionalities provided by this framework are based upon certain coding standard. It is an outcome of many years of experience.





* 1. Benefits of using Collection Framework
     1. Reducing Programming effort
     2. Performance of the program is improved because we are using standard code.
     3. Increase Productivity

1. List:
   1. Maintaining the insertion order of the data
   2. Allow duplicate elements
   3. Collection Types supported by List Interface
      1. ArrayList
      2. LinkedList
      3. Vectors
2. Set:
   1. Use Hashing, LinkedList and Tree storage mechanism
   2. Does not allow duplicate elements
   3. Maintains no order, LinkedHashset maintain insertion order
   4. Collection Types supported by Set Interface
      1. HashSets
      2. LinkedHashSet
      3. TreeSet
3. Exception Handling

// Error During the Compilation :: Compiler is taking care of it, As a programmer we are not responsible to handle this

// Logical Error :: We have to implement correct logic to avoid this error

// Error while running the program :: When we get the error during the execution of the program,

// Program stops working and Program gets terminated. System Failure

// Various Situations in which your program may throw exceptions during the execution

// Invalid input

// Unavailability of resources like (file, Database, Network connect is not there)

// Trying to perform the incorrect calculation

// Trying to access unavailable references

// Memory not available (Overflow, Underflow)

// To handle Exception Java Provides

// try block

// catch() method

// finally block

// throw keyword

// throws keyword

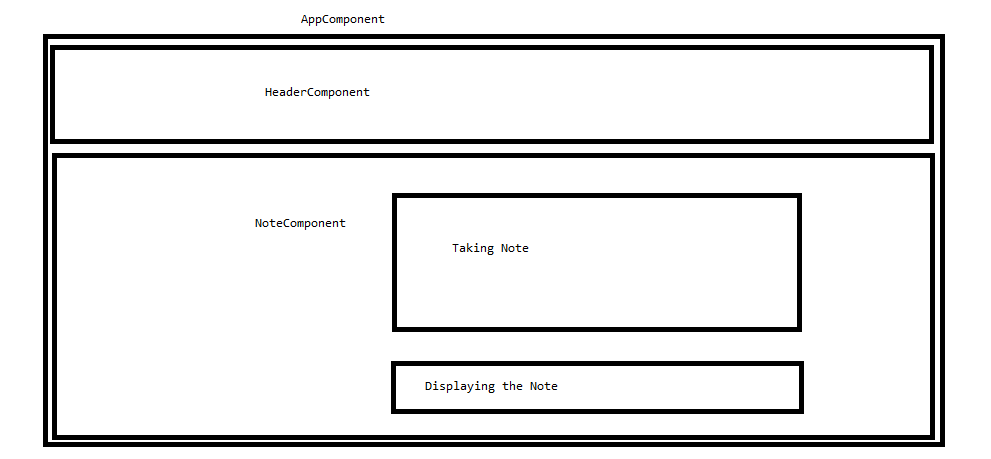
// Nested Try Catch

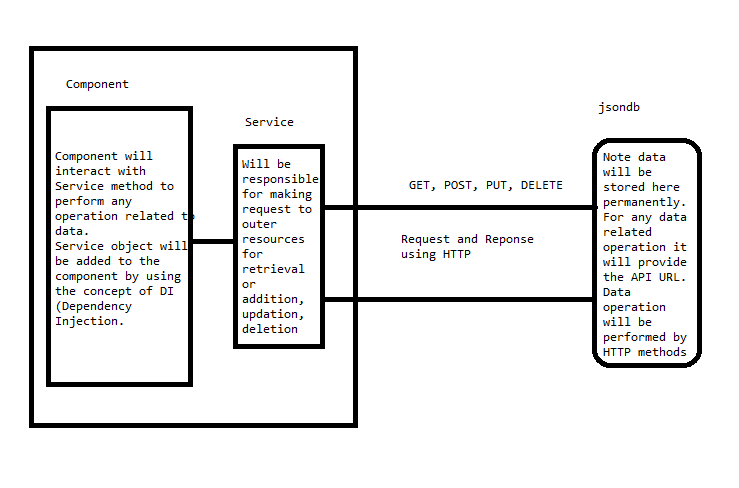
// When you entire logic is divided into multiple sub-logics

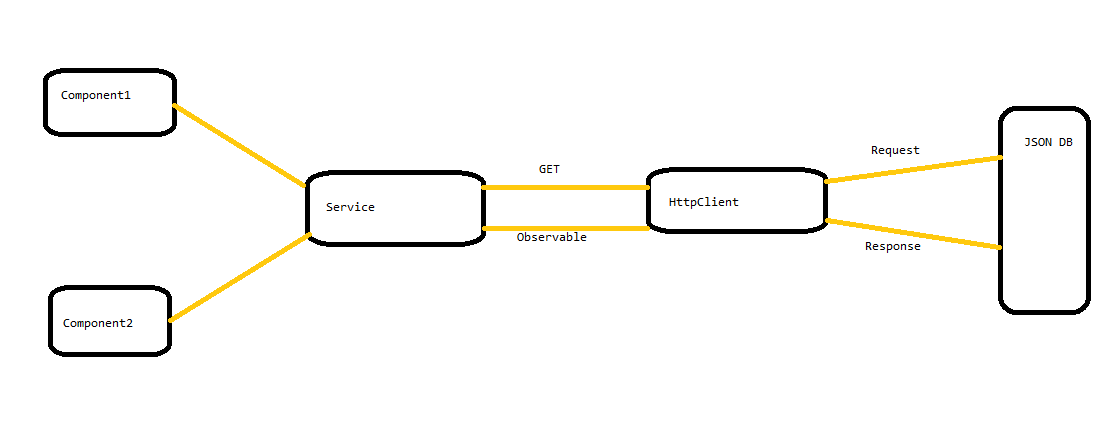
// For entire logic there can be some common exceptions

// But for sub logics there are another exception which can be handled separately

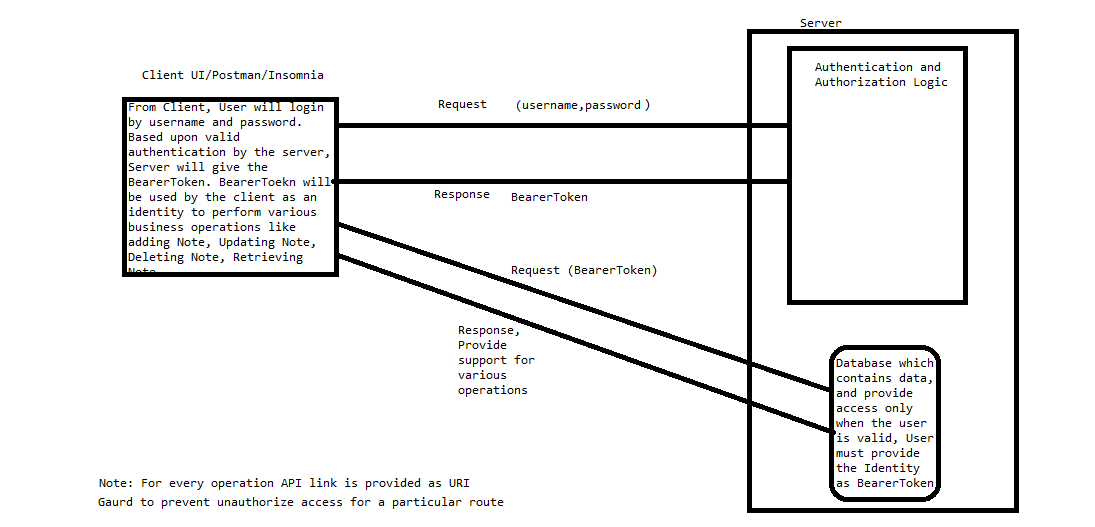
// Reading a data and displaying them from file. But before reading I also want to check the validity of file, length of file and permissions related to file



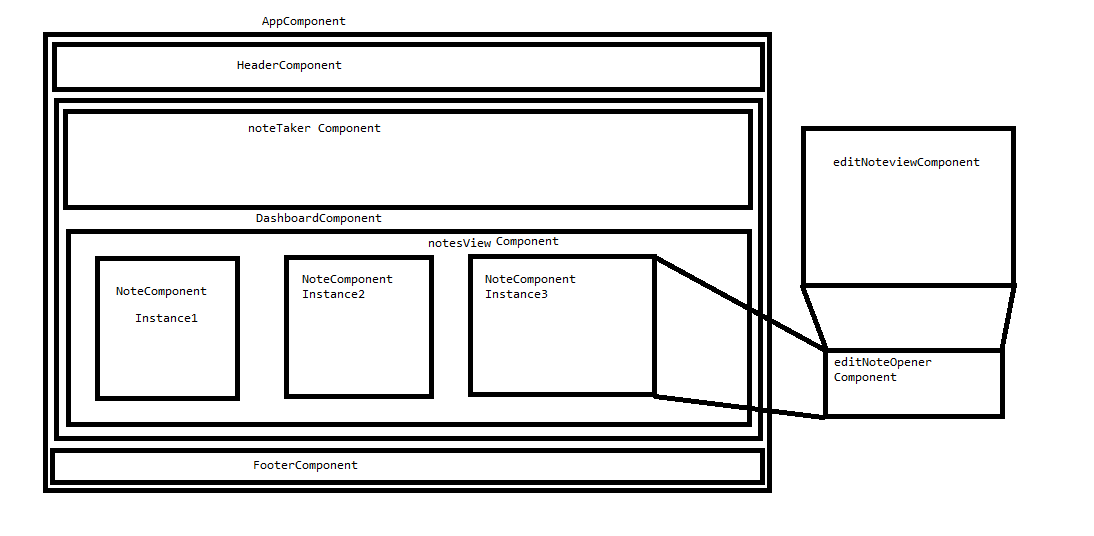


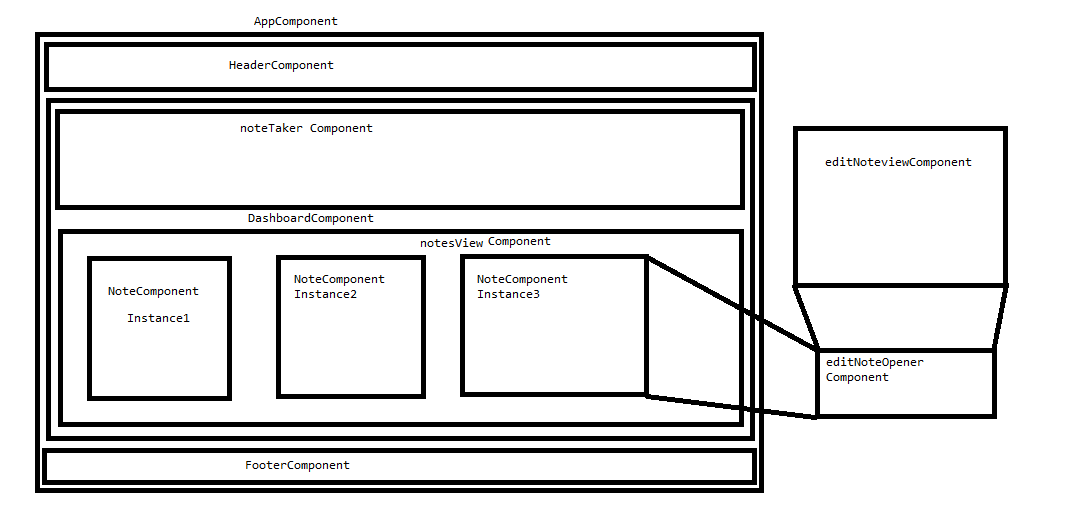


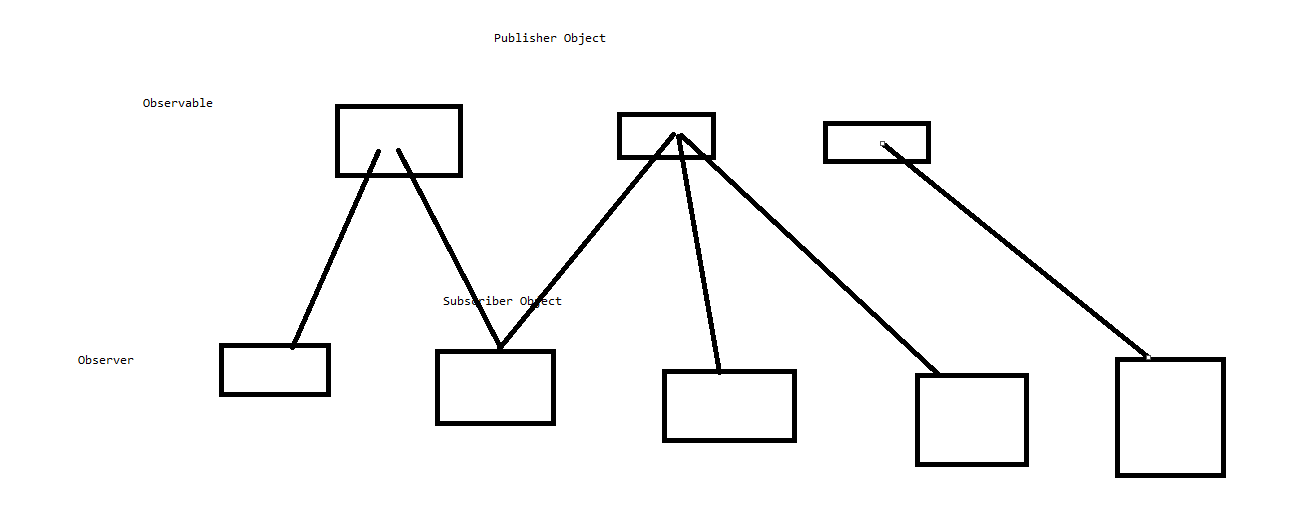
1. Why do we need form?
   1. Data Collection
   2. Display Dashboard
2. Different types of Forms in Angular? How they Differ from each other?
   1. Template Driven
      1. Entire implementation is controlled through .html file
      2. We use many tags or directives for the Form implementation
      3. Validation of data is done inside .html page by using directives
      4. We need to import formsModule
      5. We can implement simple input forms just to accept or display the data
      6. Asynchronous
   2. Reactive
      1. Entire implementation is controlled through .ts file
      2. We use classes for the Form implementation
      3. Validation of data is done inside .ts file using classes like (Validator)
      4. We need to import ReactiveFormsModule
      5. If we need to manage the state of data through the form then we use this form, It is also called as Data Driven form
      6. Synchronous
   3. Different states of the form and its fields
      1. $pristine No fields have been modified yet
      2. $dirty One or more have been modified
      3. $invalid The form content is not valid
      4. $valid The form content is valid
      5. $submitted The form is submitted
   4. Form Fields
      1. ng-pristine No fields has not been modified yet
      2. ng-dirty One or more fields has been modified
      3. ng-valid The form content is valid
      4. ng-invalid The form content is not valid
      5. ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
      6. ng-invalid-*key* Example: ng-invalid-required
3. Angular Routing
   1. Provide the support for implementing navigation
   2. In Agular we call navigation as routing
   3. Angular provides many classes and modules to supporting Routing
   4. Routing can be implemented in mainly two ways
      1. Static Routing
         1. When the route is fixed from a particular link
      2. Dynamic Routing
         1. When it is not fixed, route is decided based upon the data or outcome of particular actions
   5. To implement routing we need to import RouterModule
   6. Till angular6 we were writing the routing definition inside app.module.ts file or main module file
   7. From Angular7 onwards a separate file is getting created when you choose routing support while creating the application.



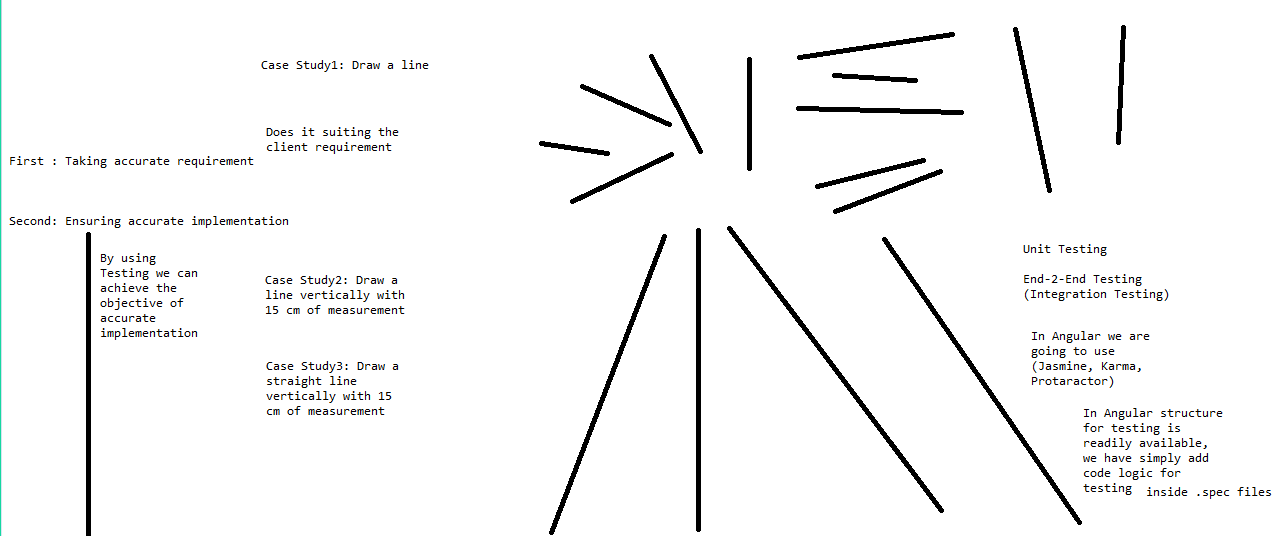
1. Component Interactions



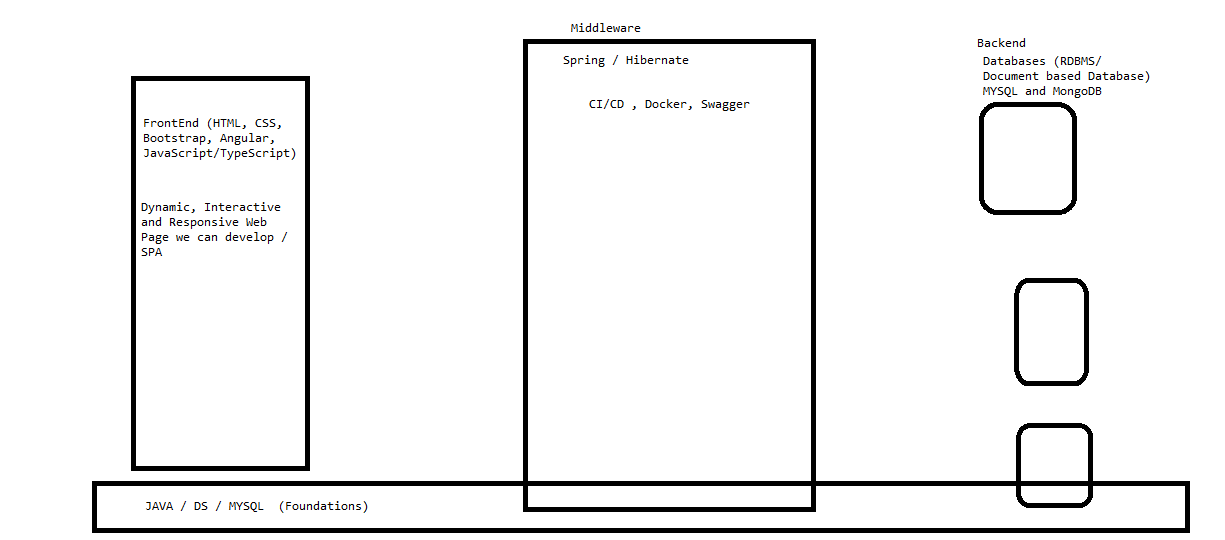




1. Angular Testing

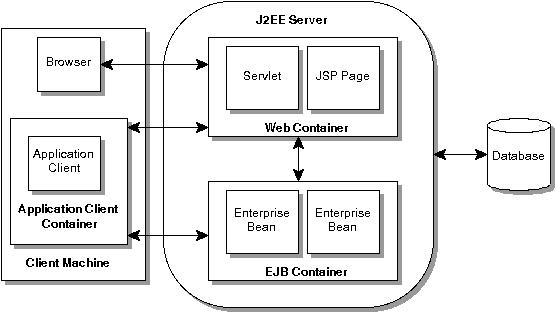


1. In today’s session we are going to start with Spring (Java Technology)



* 1. User SignIn and SignUp functionalities
  2. All operations related to Note management
  3. With Note there will be some additional information
     1. Category of Note
     2. Reminder for Note
  4. Search functionality for Notes
  5. Different views in which note information to be displayed.

1. We are going to develop Enterprise Application with all the functionalities
   1. What is Enterprise Application?
      1. MS Office is software and also solving the need of the organization but this not enterprise application.
      2. Large
      3. Data Centric
      4. Secure
      5. Can be used across many platforms and corporate networks (Internet, Intranet)
      6. User friendly
      7. Must Satisfy the stringent requirement
      8. Business Oriented
      9. Mission Critical
   2. J2EE (Java 2 Enterprise Edition) (Servlet and JSP, EJB)



* 1. Problems with earlier Enterprise Applications
     1. Complex Structure
     2. Heavy Hardware requirement
     3. Code were complex
     4. Many design patterns were used
     5. Consuming lots of resources
     6. Response time were high
  2. To overcome with above issues There is Entry of Spring kind of technologies



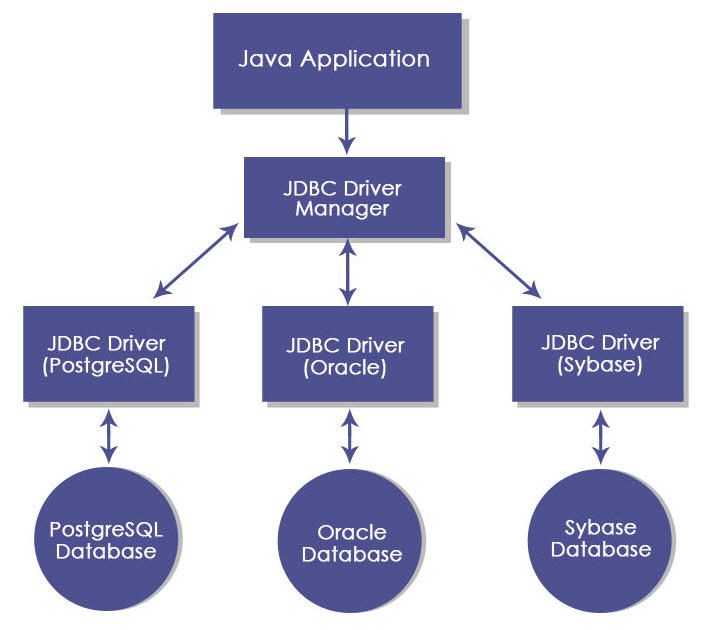


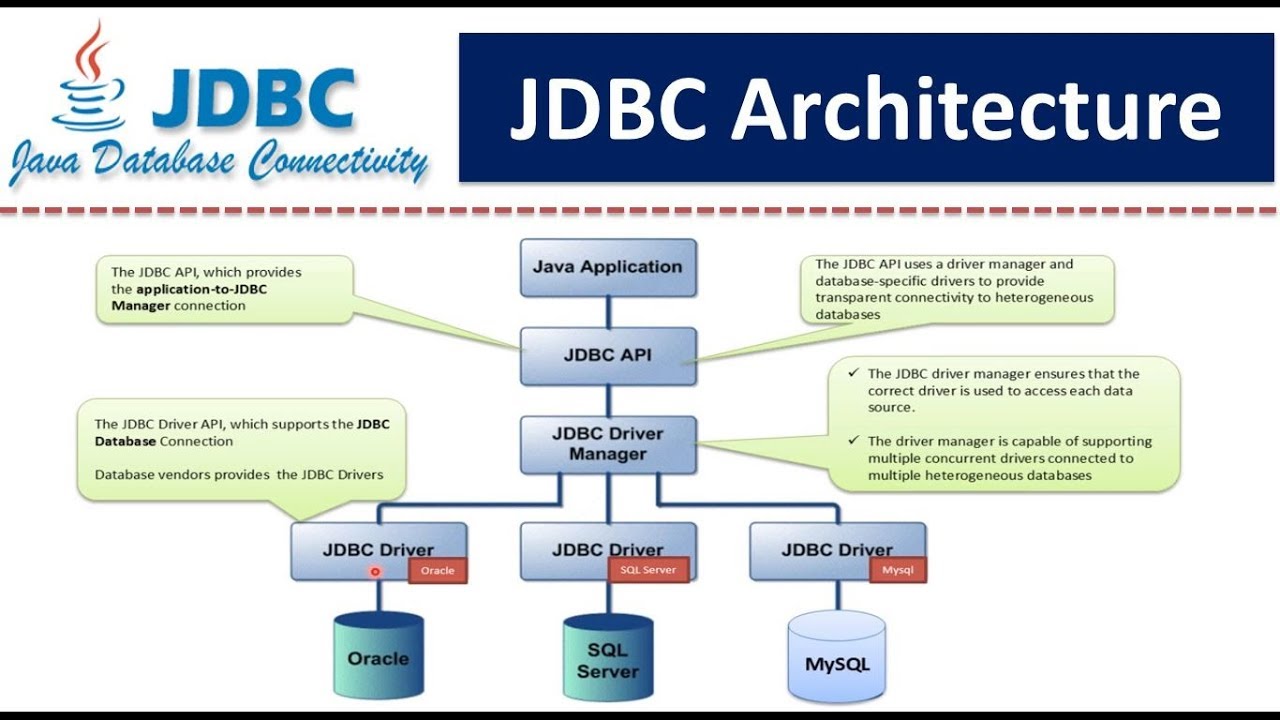
1. To work with Spring, we need
   1. Java
   2. Eclipse / STS
   3. Database
   4. Maven repository
2. So far what all we have covered in Spring
   1. Spring Core
      1. XML based implementation
      2. Annotation based implementation
      3. IOC (DI)
   2. Spring MVC
      1. Standard MVC Model
      2. Spring MVC



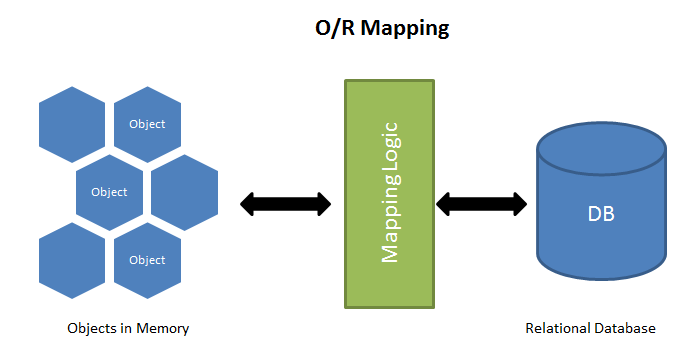


1. Today’s Agenda
   1. Spring MVC supported by Hibernate and Database we are going to use is MYSQL for storing the data.
      1. JDBC (Java Database Connectivity)
         1. It helps to make communication possible between Java Application and Database System Application





* + - 1. Steps which to be performed for JDBC Implementation inside Application
         1. Register the Driver
         2. Establish the Connectivity with the Database and open for use
         3. Send the query to perform various operations to the Database through your Java Application
         4. Read the result generated by the Database and give it to user.
    1. Hibernate
       1. It is an ORM (Object Relation Mapping) tool

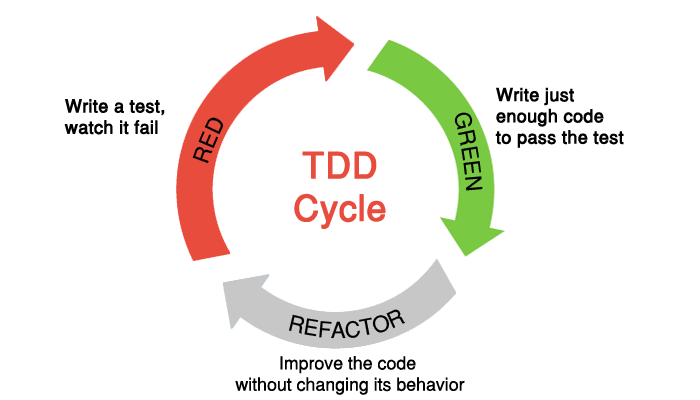


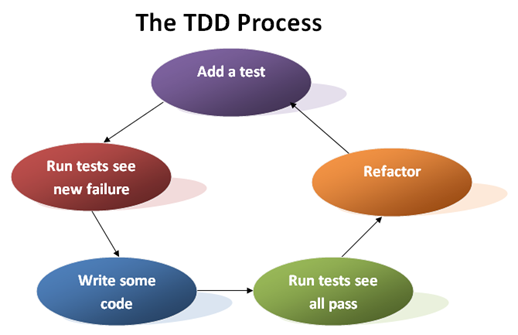


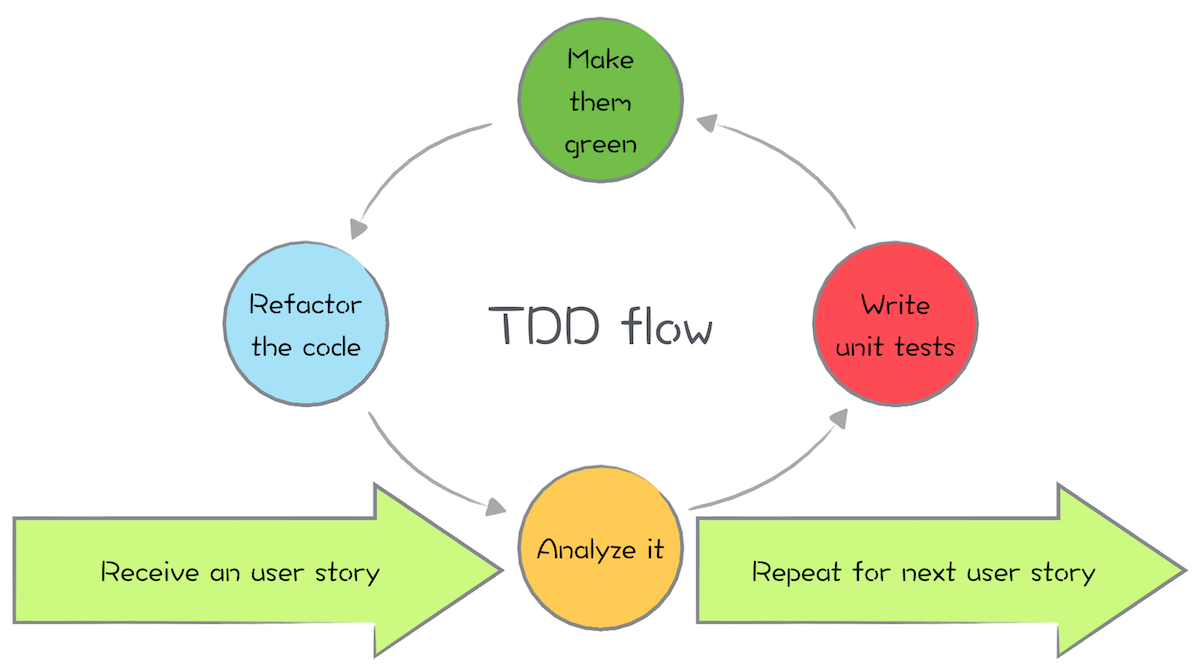
MYSQL 🡪 SQL

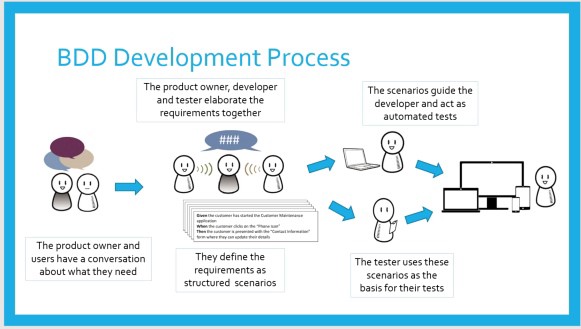
In case of Hibernate are we querying the database -> We are querying Objects and for that we have been provided HQL (Hibernate Query Language). EJB (JPQL)

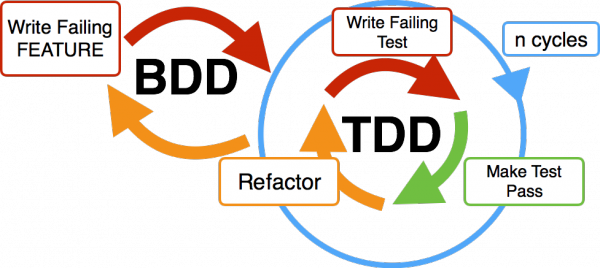
* 1. We are going to understand Spring Boot and will convert the same demo in Spring Boot. Hibernate and MYSQL
  2. Spring Boot MongoDB











1. MongoDB: Document based data store
   1. Different ways of managing the data
      1. Structured
         1. RDBMS
      2. Semi Structured
         1. XML
      3. Unstructured
         1. NoSQL
   2. MongoDB acronyms
      1. Database 🡪 Database
      2. Table 🡪 Collection
      3. Record 🡪 Document
      4. Field 🡪 Field
      5. It supports the data in Bson (Like JSON)

