Name: Shreeja Shrijan Shrestha.
Roll no: 181640
Shift: Morning.

EAPD ASSIGNMENT.

1. What is enterprise architectural framework? What are its different types? Describe TOGAF and Zachman framework in details.

→

An enterprise architecture (EA) is a conceptual blueprint for conducting enterprise analysis, design, planning, and implementation, using a comprehensive approach at all times, for the successful development and execution of strategy. An enterprise architectural framework (EA framework) defines how to create and use an enterprise architecture, to which provides principles and practices for creating and using architectural description of a system.

The industry standard enterprise architectural frameworks include three different types of frameworks:

- Template framework - Zachman framework
- Content framework i.e, The Open Group Architecture Framework (TOGAF), the DoD Architecture Framework (DoDAF), the variants (MODAF and NAF)
- Unified Profile for EA frameworks from OMG (UDPM, UAF).

The description of TOGAF and Zachman framework is done below:

→

i) The Open Group Architectural Framework (TOGAF)

→ The TOGAF was first developed in 1995. created and owned by The Open Group, which was based on the Department of Defense's Technical Architectural Framework for Information Management. This is one most common framework that accounts for over 80% of the entire business framework structure.

The TOGAF framework consists of six important parts:

- → Architecture Development Method (ADM)
- → ADM Guidelines and Techniques
- → Architecture Content framework
- → Enterprise Continuum & Tools.
- → TOGAF Reference Model.
- → Architecture Capability Framework

TOGAF is often viewed as more an overarching process. The details & methods contained within TOGAF help guide businesses through any step of business organization. A key element of TOGAF is the Architecture Development Method (ADM) which specifies a process for developing enterprise architecture.

ii) The Zachman Framework

→ The Zachman framework is a fundamental structure for Enterprise Architecture which provides a way of viewing an enterprise and its information systems from different perspective and how components are related.

The rows of Zachman framework focus on describing the enterprise from six point of view of an stakeholders. These six perspective are based on english language interrogatives 'what', 'where', 'who', 'when', 'why' and 'how'. (known as WSH)

The columns of Zachman framework consists of set of artifacts that are description of the enterprise from a specific viewpoint of a group of stakeholders. The stakeholders are generally grouped as planners, owners, designers (architects), implementers, sub-constructors, users, or sometimes represented as viewpoints: scope context, business concepts, system logic, technology, physics, component assemblies, and operation classes.

The framework enables complex subjects to be distilled into systematic categories in the column header, using these six basic question (5WH). The answer to these questions will differs, depending on the perspective or audience.

	WHAT	HOW	WHERE	WHO	WHEN	WHY.
SCOPE CONTEXTS	Inventory identification	List of business processes	List of business location	List of important organization	List of events	List of business goals & Strategies
ENTERPRISE MODEL	Conceptual data/ object model	Business process model	Business Logistics System	Work flow Model	Master Schedule	Business Plan.
SYSTEM MODEL	Logical data model	System architecture model	Distributed Systems architecture	Human Interface Architecture	Processing Structure	Business Rule Model
TECHNOLOGY MODEL	Physical data/ Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
DETAILED REPRESENTATION	Data definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification.
FUNCTIONING ENTERPRISE	Useable data	Working function	Useable Network	Functioning Organization	Implemented Schedule	working Strategy

2. What are different types of JAVA EE frameworks to build enterprise application? Describe in detail about EJB.

→ Java EE frameworks are powerful tools to create complicated and broad bodies of enterprise applications with the help of Java programming language.

The list of various Java EE frameworks used in building enterprise application are as follows:

- Spring
- Hibernate
- Struts
- JavaServer Faces (JSF)
- Google Web Kit (GWT)
- Grails etc.
- Vaadin
- Blade
- Dropwizard
- Play.

EJB is acronym for Enterprise Java Bean. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

The EJB is used when:

- Application needs Remote Access.
- Application needs to be scalable
- Application needs encapsulated business.

There are 3 types of enterprise bean in java:

i) Session bean
- Session bean contains business logic that can be invoked by local, remote or webservice client.

ii) Message driven bean
- like session bean, it also contains the business logic but is invoked by passing message.

iii) Entity bean
- an encapsulates the state that can be persisted in database. It is deprecated and is now replaced by JPA.

In EJB, bean component and bean client both must be written in java language.

The disadvantages of EJB are:
- Requires application server.
- Requires only java client. For other languages client, we need to go for webservice
- complex to understand and develop ejb applications

→ The important benefits of using EJB are :

* Simplified development of large-scale enterprise level application
→ Application server / EJB container provides most of the system level service like transaction handling, logging, load balancing, persistence mechanism, exception handling, and so on.
→ Developer has to focus on business logic of application only.
→ EJB container manages life cycle of EJB instance, thus developer doesn't need to worry about when to create / delete EJB objects.

3. What is class diagram in UML? What are basic components of a class diagram? Consider any two class diagram in UML like Hotel management system & ATM system.

→ A class is a blueprint that is used to create object. UML class diagram gives an overview of software system by displaying classes, attributes, operations, and their relationships. This diagram defines an object in system and the different types of relationships that exist among them.

The essential elements of a UML class diagram are:

i) Class Name

→ The name of class appears at topmost compartment and is blueprint of objects which can share same relationship, attributes, operations and semantics. It must start with capital letter.

ii) Attributes

→ An attribute is named property of class which describes the object being modeled. It lies below name compartment.

iii) Operation

→ Operations are list of behavior that lies in the third compartment of class diagram.

→ Relationships are basically of 3 types in class diagram:

i) Dependency

→ It is relation between two or more class in which one class depends upon another meaning changes in one class may force changes in another.

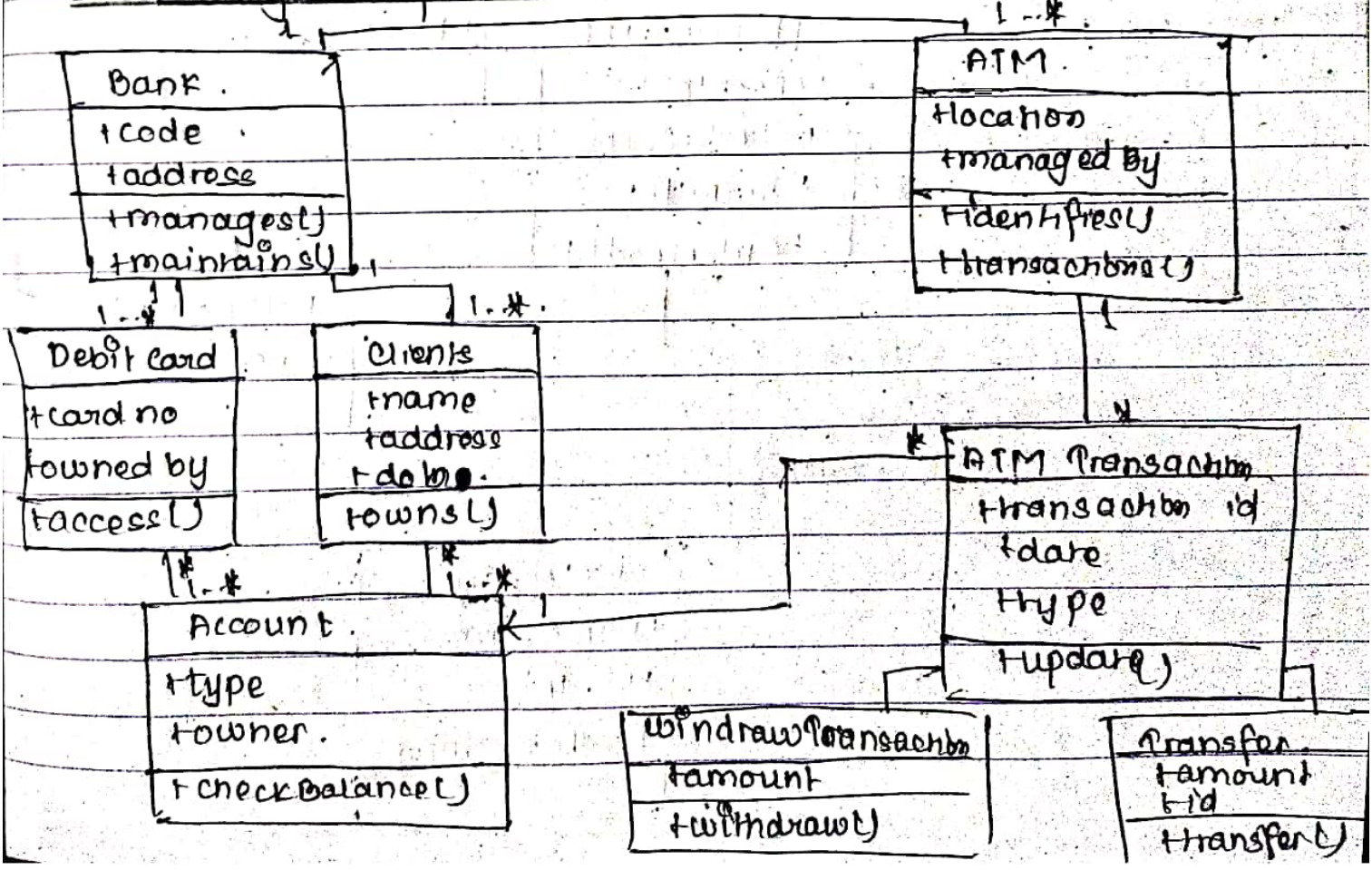
ii) Generalization

→ A generalization helps to connect a subclass to its superclass where subclass is inherited from superclass.

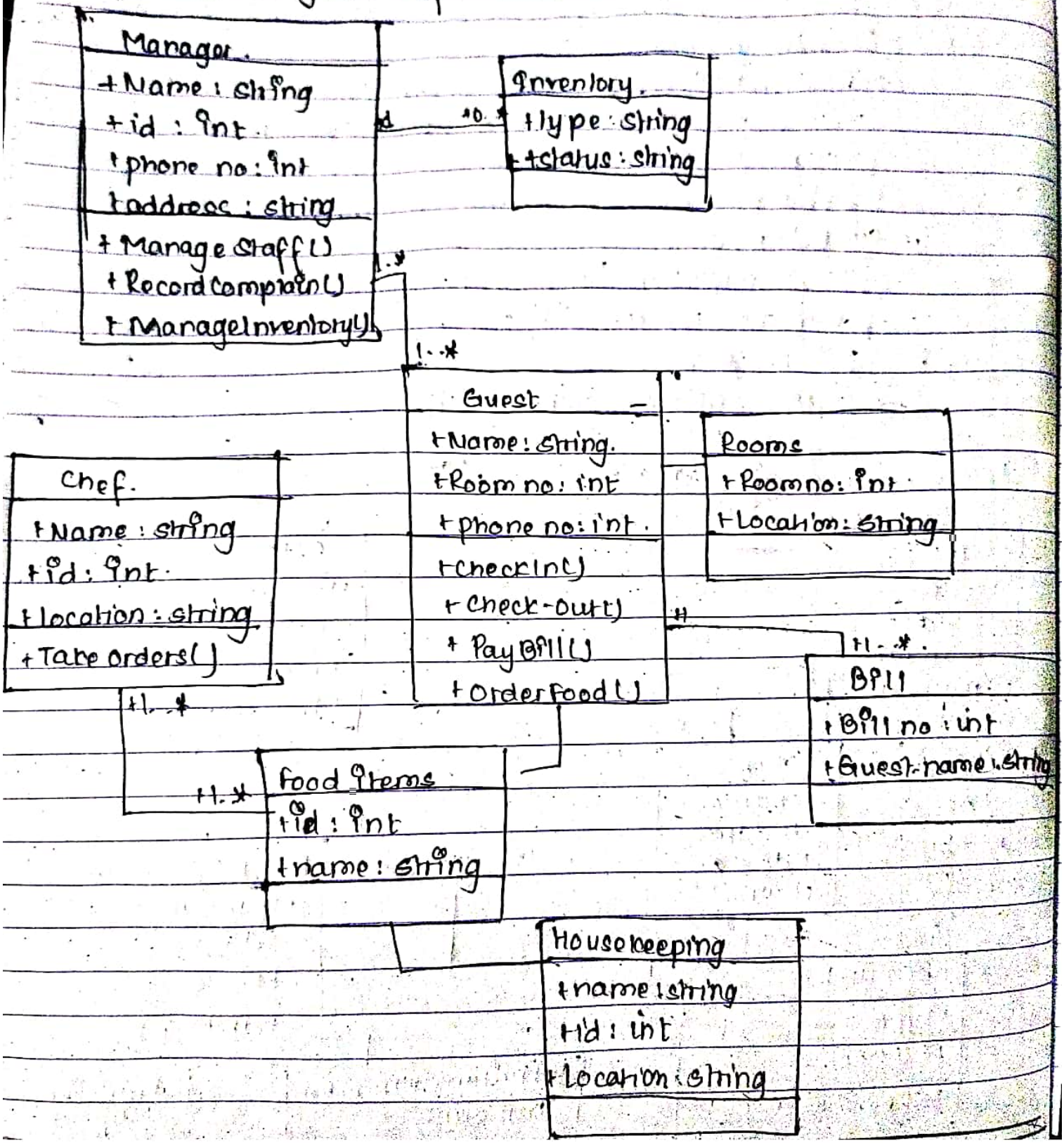
iii) Association

→ This represents static relation between two classes.

Class diagram of ATM:



Class diagram of hotel management system



4. Within overall Agile environment, what is the purpose of BDD?

→ Agile environment is simply a company or an organization that creates and supports Agile project management this means that the company culture encourages project teams to adopt the principles and values of the Agile methodology. The 4 pillars that any agile environment needs to support are:

→ Individuals and interactions over process & tools
→ Working software over comprehensive documentation
→ Customer collaboration over contract negotiation
→ Responding to change over following a plan.

→ Behavior Driven Development (BDD) is a synthesis and refinement of practices stemming from TDD (Test Driven Development) and ATDD (Acceptance Test Driven Development). It is a process designed to aid the management and the delivery of software development projects by improving communication between engineers and business professionals thus ensuring all development project remains focused on delivering what the business actually needs while meeting all the requirements of the users.

within an Agile environment

In general, BDD, using user stories, describe clients desired behavior of the planned software for each of the relevant roles. And it is achieved by user roles rather than functional grouping. The general assumption

that different users within the organization will do different things with the information. Some will capture the information while others will analyze the information and use it in reporting it to the next level of organization.

Behavior Driven Development (BDD) is ~~first~~ test first, agile testing practise. In an agile environment it provides built in quality by defining tests before, as a part of specifying system behavior. It creates collaborative and shared understanding of requirements between the business and Agile Teams. It helps to guide development, decrease rework, and increase flow.

For eg:
Let us consider an emergency relief response team that wants to use data system to efficiently serve the locals impacted by a series of tornados.
The users here can be:
- → Field Agent
- → Area Manager
- → State Manager

User stories can be:
- As field Agent, I want to capture number of people cared for every hour and service being provided. All the data should be in tabular form so I can manage activities of shelter.

→ As a disaster manager, I want to be able to list all of the shelters, showing the name of each shelter manager, the name of each shelter, the number of people in each shelter, and the last delivery of water to each shelter so I can report this information to the State Disaster Manager

→ As a State Disaster Manager, I want to be able to create a report that combines all the data from the Area Disaster Manager's report so that I can report to the National Disaster Manager.

Q. What is continuous integration? What do you mean by integrated testing?

→ Continuous Integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. The CI process is comprised of automatic tools that assert the new code's correctness before integration.

CI is a valuable asset to a software producing organization. CI enables better transparency and insight into the process of software development and delivery. Some of the benefits of CI are that it enables scaling, it improves the feedback loop enhancing communication between collaborators.

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of testing is to expose the interaction between integrated units. In this testing stage, test drivers and test stubs are used.

Integration testing is the second level of testing performed after unit testing and before system testing. This testing is done by developers themselves or some independent tester.
The various approaches that can be used for integration testing are:

i) Big bang
→ Big bang is integration testing approach where all or most of the units are combined together and tested at one go. This approach is taken when the testing team receives the entire software in a bundle.

ii) Top down
→ In this approach, the top level units are tested first and lower level units are tested after that. Test stubs are needed to simulate lower level units which may not be available during initial stage.

iii) Bottom Up
→ This approach tests bottom levels first and upper units

step by step after that. This approach requires test drivers to simulate higher level units.

iv) Sandwich / hybrid

→ It is the combination of bottom up and top down approaches.

The advantages of this testing procedure are:
- It makes sure that the integrated modules/components work properly.
- It detects error related to the interface.
- Integration testing can be started once the modules to be tested are available. It can replace unavailable modules with stubs and drivers.

---

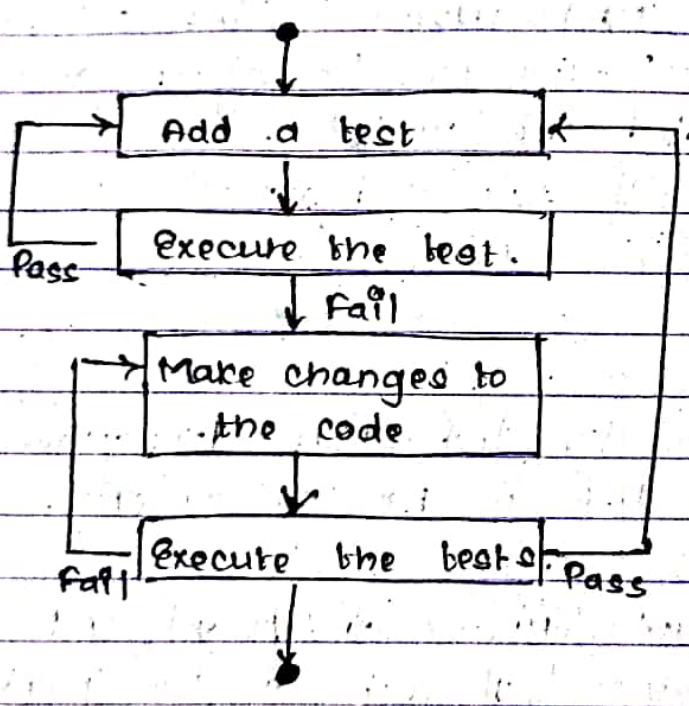
6. What do you mean by test driven development? How would you implement TDD in real life scenario. Give some examples.

→ Test Driven Development (TDD) is a software development process that relies on the repetition of very short development cycles: first the developer writes an initial automated test case that defines a desired improvement or new function, then produces minimum amount of code to pass that test, and finally refactors the new code to acceptable standards.

Test Driven Development helps to make code clearer, simple, bug free. This methodology helps to avoid duplication of code as we write them in small amount in order to pass tests.

General steps followed in TDD are as follows.

- → Add a test.
- → Run all tests & see if any new test fails.
- → Write some code.
- → Run tests and refactor code.
- → Repeat.



The failure determines where the developers' effort will be spent. Only requirements with failed test cases will be addressed with respect to coding.

Each TDD cycle defines to write a test, make it run, change the code and repeat process.

A real life example of TDD can be ~~for~~ as follows:

eg:

Create a class 'Password' that will satisfy the following conditions.

1) The password should be between 5 to 10

Code written:

```
public class TestPassword {
    public void TestPasswordLength() {
        PasswordValidator pv = new PasswordValidator();
        Assert.assertEquals(true, pv.isValid("Abc123"));
    }
}
```

Scenario 1: To run test PasswordValidator is created.

```
public class PasswordValidator {
    public boolean isValid(String Password)
    { if (Password.length() >= 5 && Password.length() <= 10)
        { return true;
        }
        else
        { return false; }
    }
}
```

Output: PASSED

→

Scenario 2: There is no need of creating an instance of class PasswordValidator. Thus, it is removed.
So;

```
public class TestPassword {
    public void TestPasswordLength() {
        Assert.assertEquals(true, PasswordValidator.isVa
                                    ("Abc123");
    }
}
```

Scenario 3: After refactoring the output shows failed status.

The advantages of TDD are:
- Early bug notification.
- Better designed, cleaner and extensible code
- Confidence to refactor
- Good for teamwork
- Good for developers.

7. What is continuous integration? Define unit and integrated testing.

→ Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check is then verified by an automated build allowing teams to detect problems early.

The continuous integration process is comprised of automatic tools that assert the new code's correctness before integration. A source code version control system is the crux of continuous integration. It helps to scale up headcount and delivery output of engineering teams.

The benefits and challenges of continuous integration are as follows:

- Enabling Scalability
- Improve the feedback loop
- Enhance communication
- Adoption and Installation
- Technology learning curve.

Unit testing is level of software testing where individual units/components of a software are tested. The purpose is to validate that each unit of software performs as designed. Unit testing is the first level of software testing and is performed prior to integration testing.

The benefits of unit testing are:

- It increases confidence in changing/maintaining code.
- Development is faster.
- The cost of fixing a defect detected during unit testing is lesser compared to defects detected at higher levels.
- Debugging is easy.

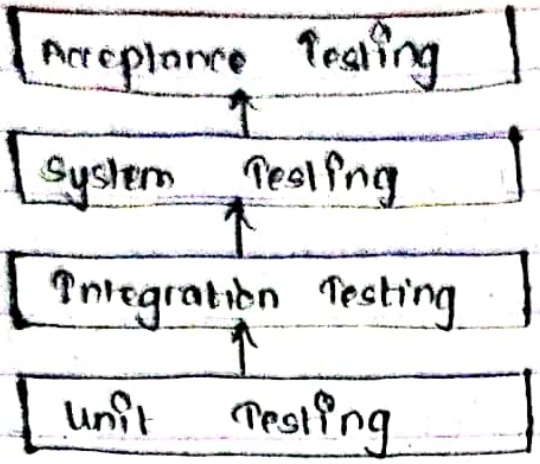


fig: Level of software testing

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this testing is to expose faults in the interaction between integrated units. Test driver and test stubs are used.

Various approaches used in integration testing are:

- Big bang
- Top down
- Bottom Up
- Sandwich / Hybrid

The advantages of integration testing are:

- It exposes defects in the interfaces
- Makes sure that the integrated modules work properly.

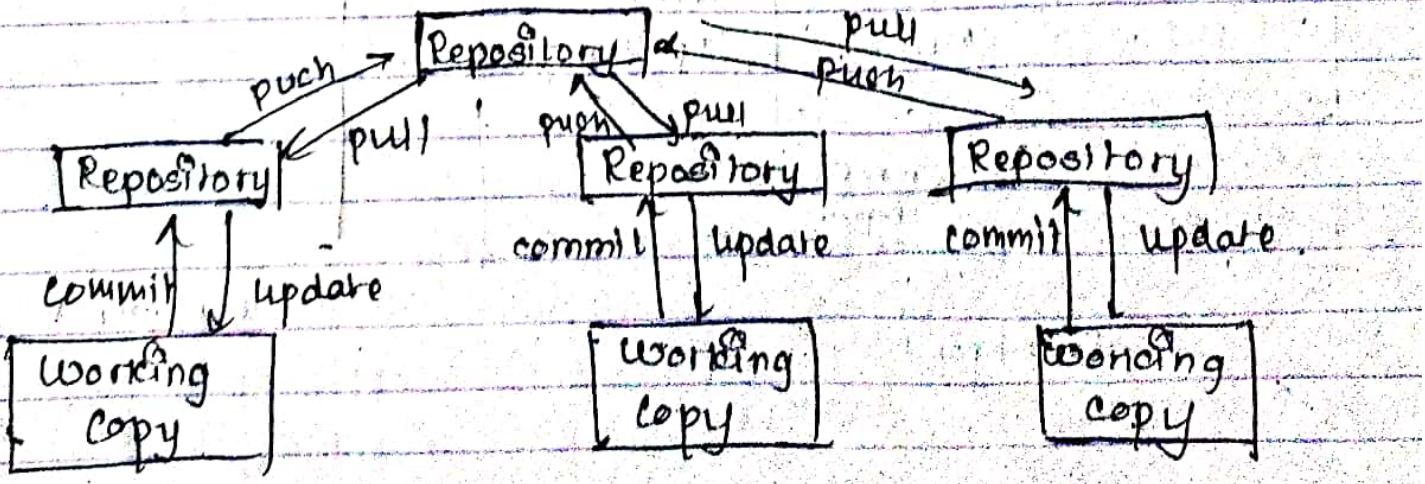
& What is version control system? Define distributed version control system. Why git and repository? How GitHub works.

→ Version control system are category of software tools that helps record changes to file by keeping a track of modifications done to the code.

Distributed version control system contains multiple repositories. Each user has their own repository and working copy. Just committing changes will not give others access to your changes. This is because commit will reflect those changes in local repository and to make them visible they are to be pushed to central repository

To make changes visible to other, 4 things are required:

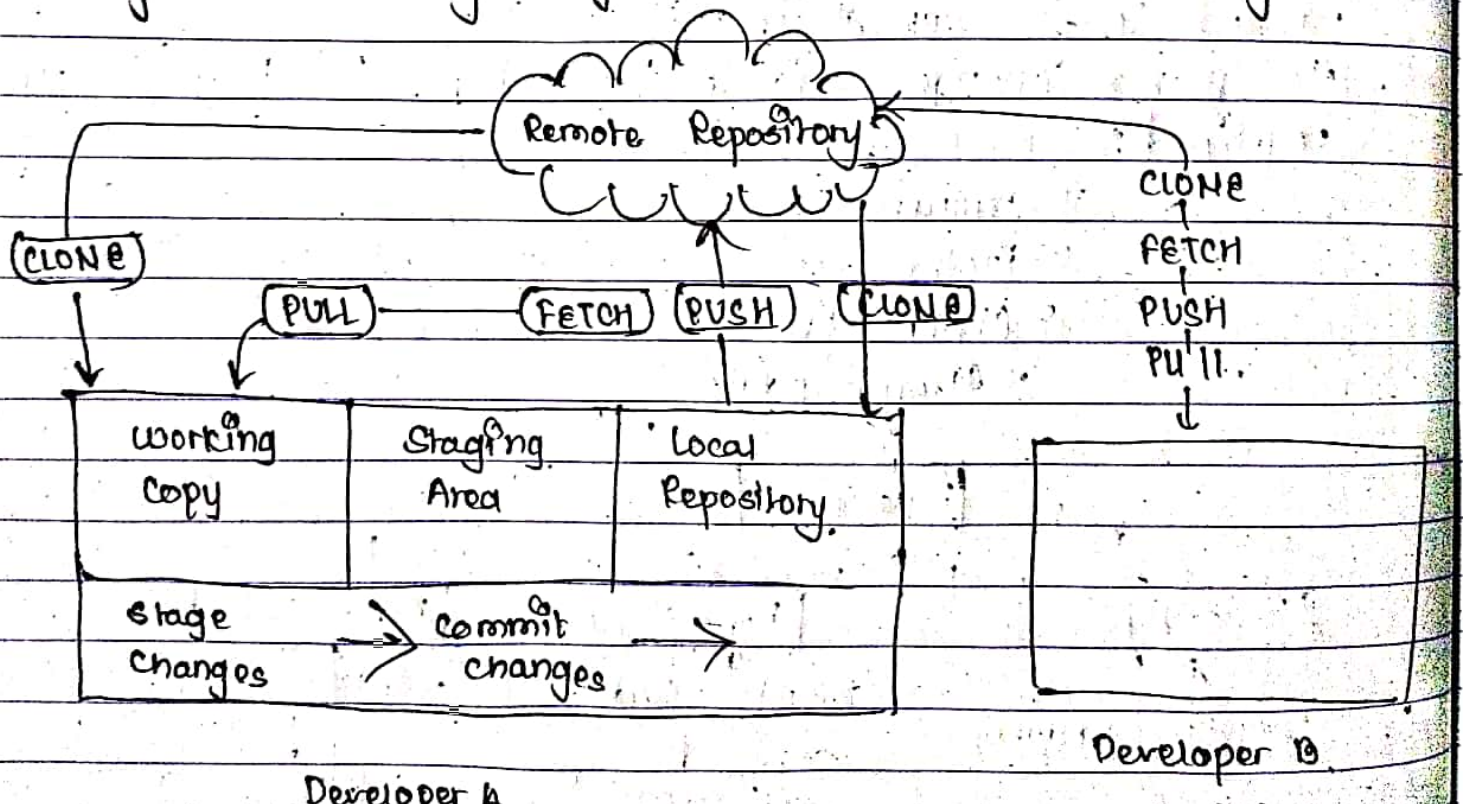
- commit
- Push
- Others pull
- They, update



A repository is database of changes. It contains all the edit and historical versions (snapshots of the projects).

Git is a distributed version-control system for tracking changes in source code during software development. ~~Here~~ It is free and open source version control system, originally created by Linus Torvalds in 2005. Unlike older centralized version control system system, Git is distributed. The files for software projects are stored in a repository.

Github is a service for hosting software repositories managed by the Git version control system.



Github allows us to share coding projects of all scale with the world with its amazing features. To use github we initially create a new repository in github. Then, we copy address to clipboard and clone the project in our computer with the command:

git clone <repo address>

Once the folder is cloned, it will appear in that very location of PC. This is local copy of remote repository. Now, we can add files and update in local repo, and then push it up to Github.

---

9. What is dependency injection and inversion of control? What are the different types of dependency injection? Implement dependency injection using Spring boot or any platform.

→ Dependency injection is a pattern through which IoC is implement. It is the act of connecting objects with other objects, or "injecting" objects into other objects.

eg:

```
public class Store {
    private Item item;
    public Store() {
        item = new ItemImpl1();
    }
}
```

Inversion of control is a principle in software engineering by which the control of objects or portion of program is transferred to a container or framework.

There are basically three types of dependency injection. They are:

1. Constructor Injection
   → They are dependencies are provided through a class constructor.
2. Setter Injection
   → The client exposes a setter method that the injector uses to inject the dependency
3. Interface Injection
   → The dependency provides an injector method that will inject the dependency into the client passed to it. Client must implement an interface that exposes a setter method that accepts the dependency.

Eg:
Let us consider an application which has a text editor component and we want to provide spell check:
Standard code would look like:

→

```
public class TextEditor {
    private SpellCheck spellChecker;

    public TextEditor() {
        spellChecker = new SpellChecker();
    }
}
```

This can be done with the help of IOC as follows:

```
public class TextEditor {
    private SpellChecker spellChecker;

    public TextEditor(SpellChecker spellChecker) {
        this.spellChecker = spellChecker;
    }
}
```

---

10. Why do we need design patterns & what are the different types of design patterns? Describe in details ~~with~~ with suitable examples of Factory Design Pattern, Singleton Pattern & Lazy initialization.

→

→ Design patterns are recurring solutions to software design problem that occur again and again in real world application development. Patterns are about design and interaction of objects, as well as providing a communication platform concerning elegant, reusable solutions to commonly encountered programming challenges.

We need design pattern because:

- → They give the developer a selection of tried and tested solutions to work with.
- → They aid communication by the fact that they are well documented.
- → They enhance reusability of code.
- → They reduce technical risk.
- → They reduce coupling.

There are basically 3 types of design pattern:

i) Behavioral pattern.

→ It describes the interaction between objects and focus on how objects communicate with each other.

ii) Creational pattern.

→ These are used to create objects for a suitable class that serves as solution for problem

iii) Structural pattern

→ They are concerned with how classes and objects are composed to form larger structures

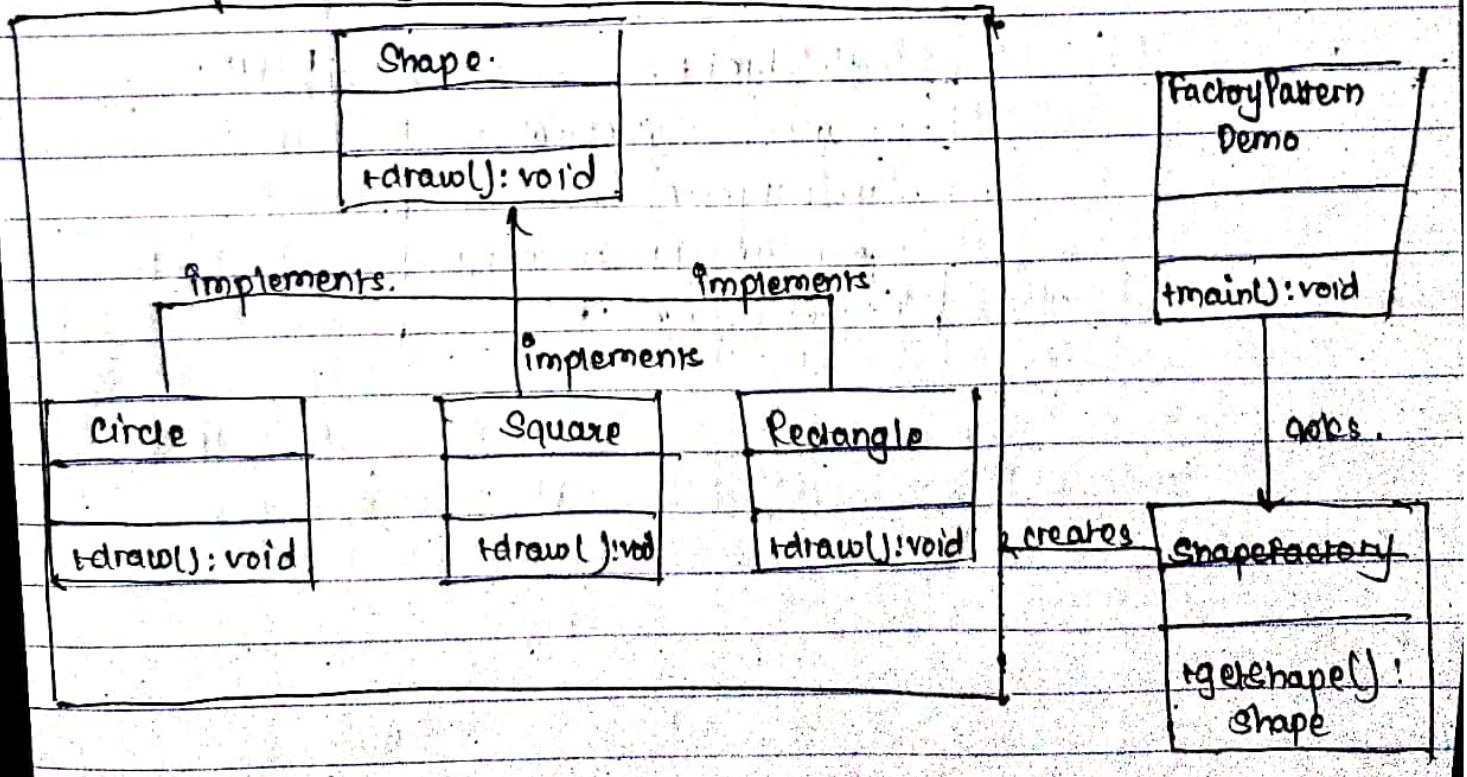
factory design pattern.

- A factory pattern or factory method pattern says that. "just define an interface and abstract class for creating an object but let the subclasses decide which class to instantiate. This pattern is also known as virtual constructor.

factory design pattern is used when:

- a class doesn't know what sub-classes will be required to create.
- a class wants ~~that~~ its sub-classes to specify the object to be created.
- pattern class chooses the creation of object to its subclass.

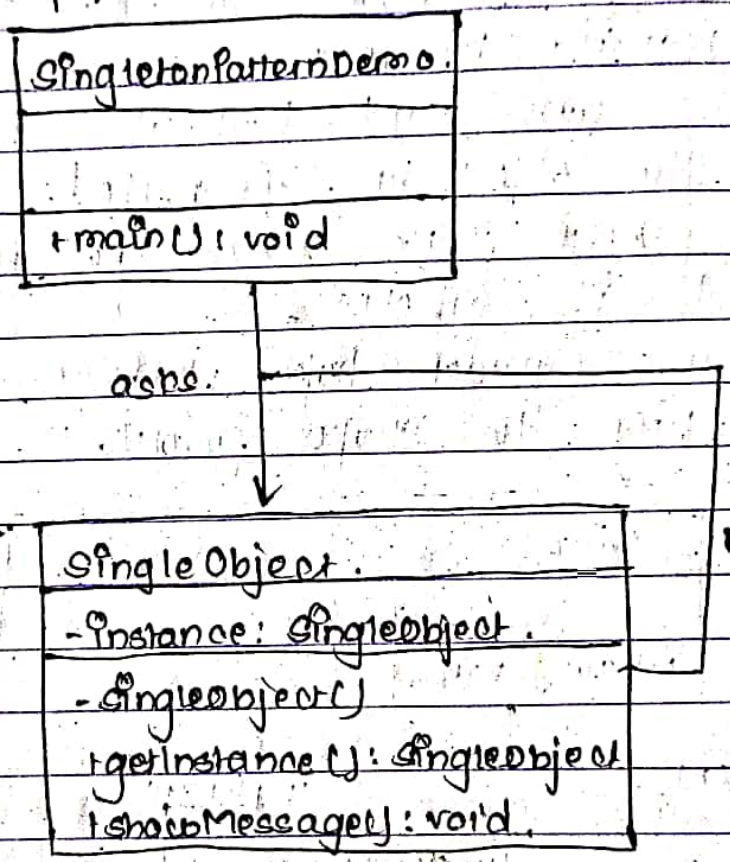
Eg: let us consider shape interface and concrete classes implementing the shape factory.



Singleton Pattern.

→ Singleton pattern is one of the simplest design patter. It is creational pattern which allows a single class to have only one object. This class provides a way to access its only object which can directly be accessed without need to instantiate the object of the class.

Eg:



Lazy initialization

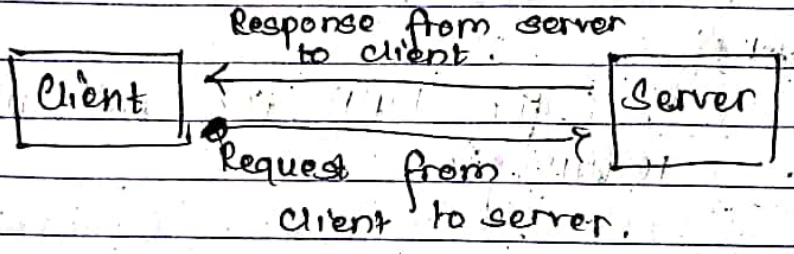
Lazy initialization is a lazy loading technique which consists of checking the value of a class field when it's being used. If that value equals to null then that field gets loaded with proper value before it is returned. In this technique the variable is initialized on its first access. Its canonical form is something like following example:

```
public fooClass foo {
    get {
        if (_foo == null) _foo = calculateFoo();
        return _foo;
    }
}
```

Lazy initialization is useful when calculating the value of the field is time consuming & we don't want to do it until we actually need the value. This is optimization technique to help responsiveness in situations where a client doesn't need the lazy initialized value.

11. What do you mean by web services? Define in detail about WSDL, UDDI, SOAP web services, and SOAP vs. REST.

→ Web services are standardized medium to propagate communication between the client and server applications on the World Wide Web. A web service is a software module that is designed to perform a certain set of tasks.



i) WSDL

→ WSDL (Web Service Description Language) is an XML format for describing network services as a set of end points operating on messages containing either document oriented or procedure oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints.

Eg of WSDL document is as follows.

```
<message name : "getTermRequest">
    <part name : "term"  type : "xs:string" />
</message>.
<message  name: "getTermResponse ">.
    <part  name: "value"  type = "xs: string" />
</message>
<portType name : "glossaryTerms">
    <operation name : "getTerm">
</port>  <input message = "getTermRequest" />.
    <output message = "getTermResponse" />.
    </operation>
</portType>
```

iii) UDDI

→ UDDI (Universal Description, Discovery and Integration) is a directory service where business can register and search for web services. It is a platform-independent framework for describing services, discovering business and integrating business services by using the Internet.

It uses WSDL to describe interfaces for web services.

Problems that UDDI specification helps to solve are:

- → Making it possible to discover the right businesses from the millions currently online.
- ↳ Defining how to enable commerce ~~online~~ once the preferred business is discovered.
- ↳ Reaching new customers and increasing access to current customers.

- Expanding offerings and extending market reach
- Solving customer driven need to remove barriers to allow for rapid participation in global Internet economy.
- Describing services and business process programatically in single, open and secure environment.

(ii) SOAP web services.

→ SOAP is an XML-based protocol for accessing web services over HTTP. It has specifications that can be used across all applications. It is also known as Simple Object Access Protocol. SOAP was developed as an intermediate language so that applications built on various programming languages could talk easily to each other and avoid the extreme development effort.

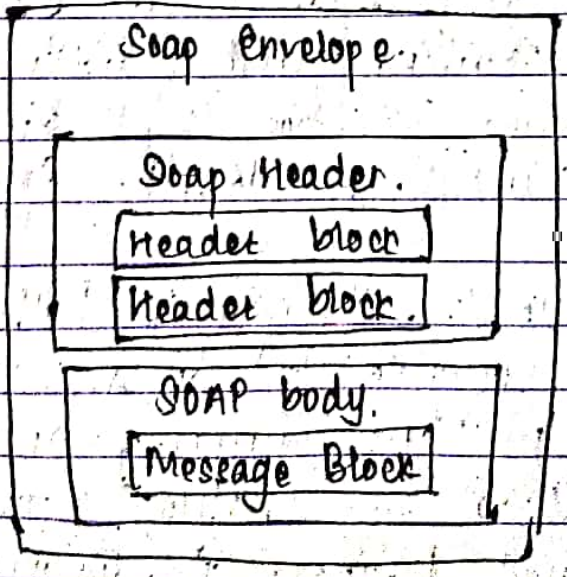


fig: Building block of SOAP.

SOAP specification defines something known as 'SOAP message' which is what is sent to the web service & the client application. An envelope elements that identifies that XML document as a SOAP message. It contains SOAP message and encapsulate all details of SOAP message & is the root element. The header element contains header information. It contains information such as authentication credentials and definition of complex types which could be used in SOAP messages. The body contains call & response information.

iv) SOAP vs. REST

→ SOAP is a protocol designed before REST and was designed to ensure that programs built on different platforms and programming languages could exchange data in easy manner.

REST was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principle of REST can be called Restful web service. A restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE

→

SOAP	REST
→ SOAP stands for Simple Object Access Protocol	REST stands for Representational State Transfer.
→ SOAP is a protocol designed with specification. It includes WSDL file which has the required information on what the web service does in addition to the location of the web service.	REST is an architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being client server, stateless, cacheable, layered system, uniform interface.
→ SOAP can't make use of REST.	REST can make use of SOAP.
→ SOAP requires more bandwidth for its usage as message contains lots of data.	REST doesn't require large bandwidth and mostly consists of JSON.

Eg:

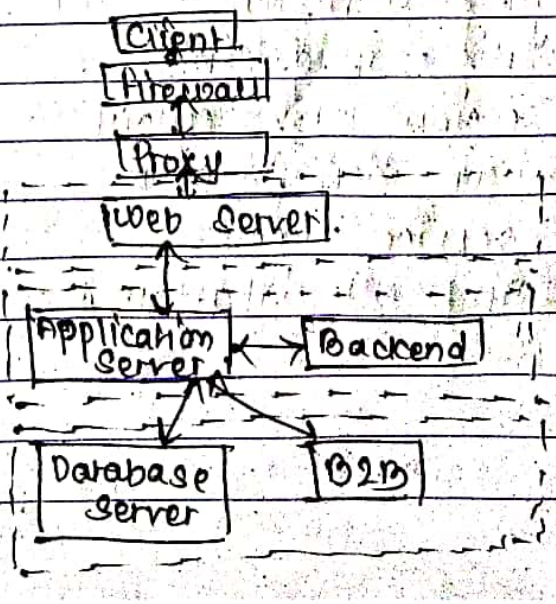
```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <SOAP:body>
    <Demo.nepwebService xmlns="http://tempuri.org/">
      <employee ID> int </employee ID>
    </Demo.nepwebservice>
  </SOAP.body>
</SOAP-ENV:Envelope>
```

Eg: {"city":"Ktm", "state":S}

12. Define layered architecture for web Application & 3 layered model: The view layer, business logic layer & data layer.

→ In a web application, with the growth of complexity the way to manage this complexity is to break the application according to responsibility and concerned. By organising code into layers, common low-level functionality can be reused throughout the application. The application can enforce restriction on its communication with other layers and helps in achieving encapsulation. Thus, layered architecture of web application is the architecture in which the application is logically seperated and physically distributed to seperate server and processes known as tiers.

The three layered model is layered architecture with three tiers namely view layer, business layer & data layer.



i) The view layer.

→ The outermost layer is the kind of model that deals with presentation layer of the content & interaction with user. It can be called view, presentation, UI. In this layer the application shows to the user what is needed to be seen and gives the tools for interaction.

ii) Business Logic Layer.

→ The central layer of the model deals with the logic of program. It receives data from upper level and transforms it, using in the inner application logics. It also retrieves data from the deepest data level and uses it to the logics. And, by integrating these two processes, it can do modifications in both levels as well.

Business logic layer contains determinant part of application logic. It includes:

- performing all required calculations & validations
- managing workflows:
  - state management
  - session management
  - user identification
  - service access
- managing all data access for presentation layer.

iii) The data layer.

→ It is the deepest layer of the 3-layer architecture; & deals with data retrieval from the source. It is an abstraction to get the plain data, that can be in wide variety of forms. It plays huge role in reusability and exchange of technologies.

---

13. What do you mean by REST API? Define in details?

→ REST is acronym for REpresentational STate Transfer. It is architectural style for distributed hypermedia system and was first presented by Roy Field in 2000 in his famous dissertation.

The guiding principle of REST are:-

i) Client-Server

→ By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms & improves scalability by simplifying server components

ii) Stateless

→ Each request from client to server must ~~co~~contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is entirely at client.

iii) Cacheable

→ Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable.

iv) Uniform interface

→ By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interaction is improved. REST is defined by four interface constraints: identification of resources; manipulation of resource through representations; self-descriptive messages and hypermedia as the engine of application.

v) Layered system

→ The layered system allow an architecture to be composed of hierarchial layers by constraining component behaviour such that each component cannot "see" beyond the immediate layer with which they are interacting.

vi) Code on demand (optional)

→ REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing no. of features required to be pre-implemented.

The key abstraction of information in REST is resource. Any information a document or image, a temporal service, a collection of other resources & a virtual information can be considered as resource. The state of resource at any particular timestamp is known as resource representation. It consist of data, metadata describing the data and hypermedia link which can help the clients in transition to the next desired state.

REST Api uses resource methods to perform desired transition. Everything that is needed to change the resource shall be part of API response for that response i.e either HTTP POST or any other and resource method doesn't necessarily mean HTTP GET/PUT/POST/Delete methods.

---

14. What do you mean by Agile & what are its different principles?

→ Agile methodology is a practice that promotes continuous iteration of development & testing throughout the software development lifecycle of the project. The four core values of Agile are:

- Individual & team interactions over processes & tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

The twelve agile principles for the methodologies that are included under "Agile Movement" are:

i) Customer satisfaction through early & continuous software delivery.
→ Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.

ii) Accomodate changing requirements throughout the development process.
→ The ability to avoid delays when the requirement or feature request changes.

iii) Frequent delivery of working software
→ Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software

iv) Collaboration between the business stakeholders & developers throughout the project.

v) Support, trust, and motivate the people involved

vi) Enable face-to-face interactions

vii) Working software is the primary measure of progress.

viii) Agile process to support a consistent development pace.
→ Teams establish a repeatable and maintainable speed at which they can deliver working software and they repeat it with each release.

ix) Attention to technical detail & design enhances agility
→ The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.

x) Simplicity
→ Develop just enough to get the job done for right now.

xi) Self-organizing teams encourage great architecture, requirement, and designs.

xii) Regular reflections on how to become more effective
→ Self improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

15) Define E-Governance Framework in detail.

→ E-governance, meaning, 'electronic governance' is using information and communication technologies at various level of government and public sector and beyond, for the purpose of enhancing governance. E-government is a generic term of web-based services from agencies of local, state and federal governments. E-governance is broader term of e-government.

E-governance concerns internally focused utilization of information and Internet technologies to manage organizational resources - capital, human, material, machines - and administer policies and procedure. The telecommunication network that facilitates E-governance is the intranet. E-governance deals with the online activities of government employees. The activities might include information to calculate retirement benefits, access to important applications, and content and collaboration with other government employees anywhere, anytime.

The interaction outside government agency is done through G2C. For private enterprises, any interaction through information system with external organizational entities occur through B2B, B2C and even C2C.

The various e-governance solution can be tabulated under various phases as following:

	External: G2C	External: G2B	Internal: G2E
Phase 1 Information.	Local/ departmental/ national Information (mission statements & organizational structure address, opening hours, employees, telephone numbers laws, rules & regulations. Petitions gov. glossary news.	Business info. addresses, opening hours, employees, telephone numbers laws, rules and regulations.	Knowledge base (Draft intranet) Knowledge management (LAN)
Phase 2. Interaction	Downloading forms on websites, submitting forms, online help with filling forms, intake permits etc. Email, newspaper, discussing groups, polls & questionnaires Personalized web pages. notifications.	Downloading form on websites. Submitting forms online help with filling forms. Intake processes for permits etc.	Email, interactive knowledge databases. Complaint handling tools.
Phase 3. Transformation	Personalized website with integrated personal account for all services.	Personalized website with integrated business accounts for all services.	Database integration