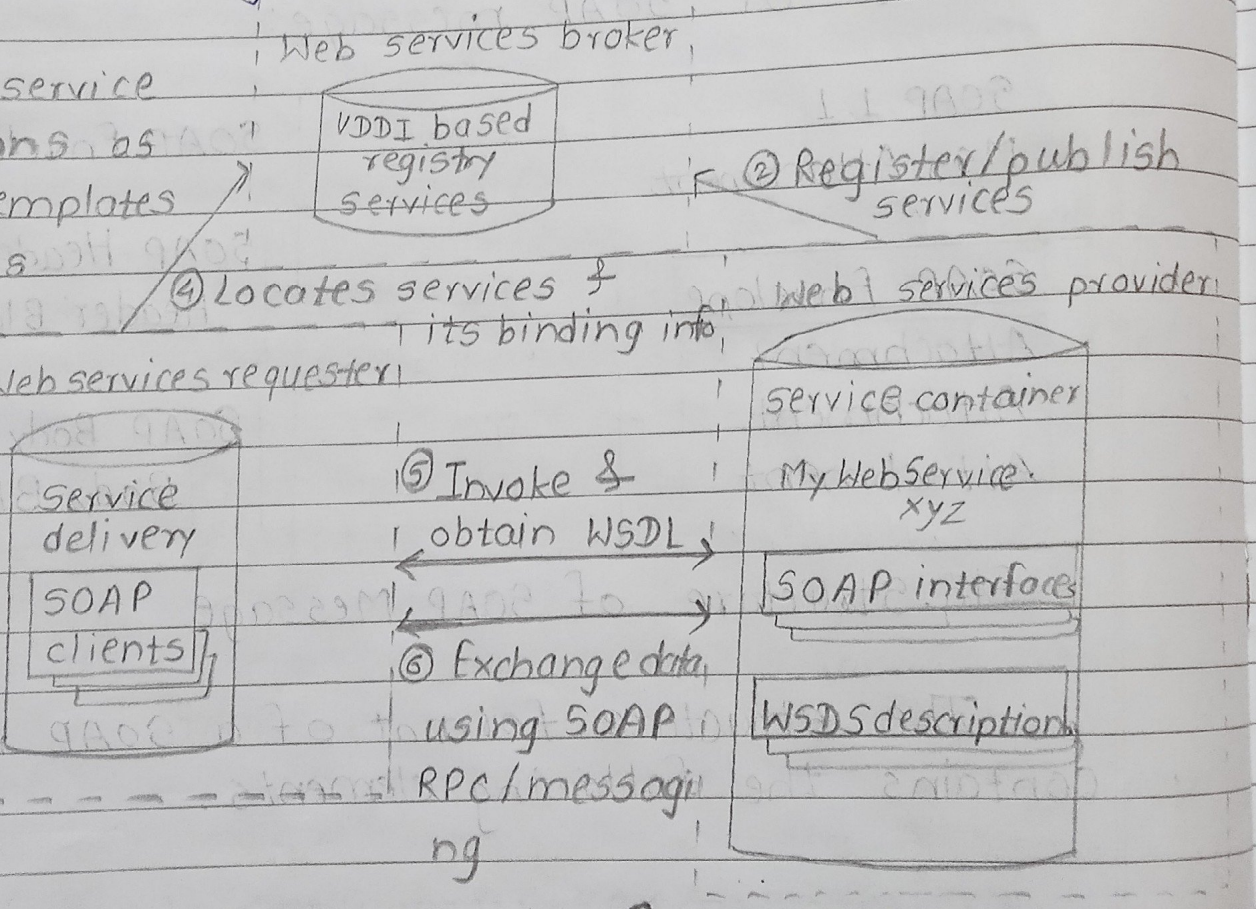
Class Test 2

1) How to implement web services?
→ We are using windows based WebSphere studio and WebSphere Application Server to develop the Pharp Web Services. The WebSphere application server resides on a Windows machine, generates a WSDL document and registers to a private

1) How to implement web service?
→ The process of implementing web services is quite similar to implementing and distributed application using CORBA or RMI. However in web services, all the components are bound dynamically only at its runtime using standard protocols.
Fig. below illustrates the process highlights of implementing web services.



ces & WSDL based service descriptions

2022/12/

The basic steps for implementing web services are as follows —

i) The service provider creates the web service typically as SOAP based service' interfaces for exposed business applications. The provider then deploys them in a service container or using a SOAP runtime environment and then makes them available for invocation over a network. The service provider also describes the web service as a WSDL-based service description, which defines the clients and the services container with a consistent way of identifying the service location, operations and its communication model.

ii) The service provider then registers the WSDL-based service description with a service broker, which is typically a UDDI registry.

iii) The UDDI registry then stores the service description as binding templates and URLs to WSDL located in the service provider environment.

iv) The service requestor then locates the required services by querying the UDDI REGISTRY.

v) Using the binding information, the service requestor then invokes the service provider & then retrieves the WSDL service description for these registered services. The service requestor then creates a client proxy application and establishes communication with the service provider using SOAP.

vi) Finally the service requestor communicates with the service provider and exchanges data

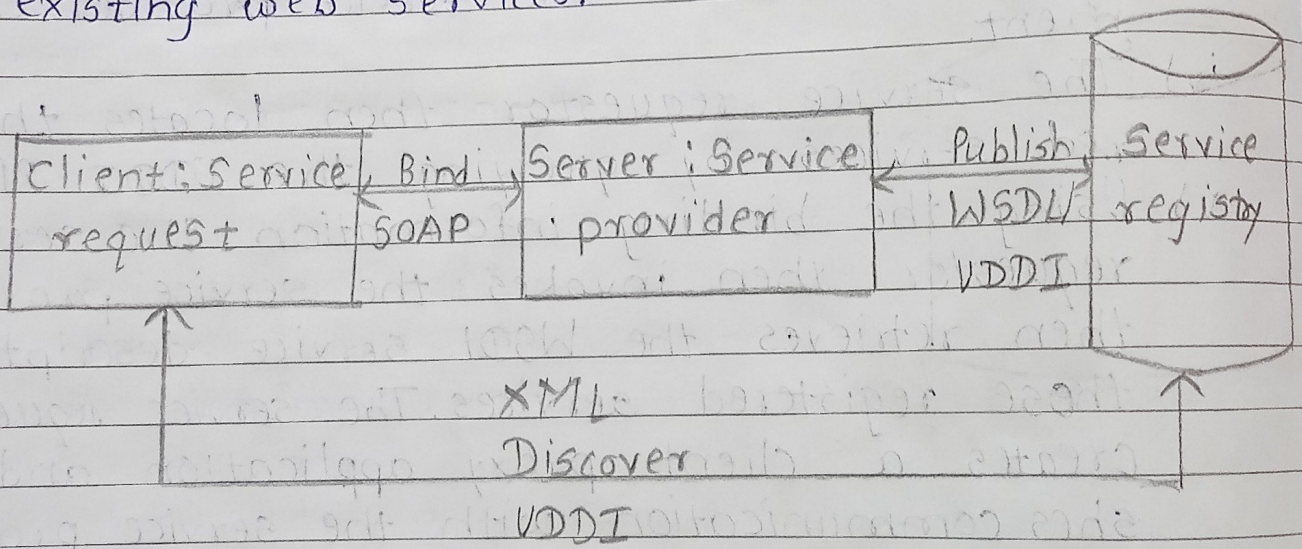
or messages by invoking the available service in the service container.

2] Explain Web Services Life cycle.

→ a) Service Provider – This role is responsible to provide the web services. The service requestor requests for the service. The service provider gives appropriate response to these requests and makes it available on internet.

b) Service Requestor – This is any client of the web service. The requestor uses the web service over the internet and sends an XML request. The client application can be a .Net, Java, or any other language-based application which initiates for some sort of web based service.

c) Service Registry – The registry is a central place where developers can publish new services. The clients can also find existing web services.



• Publish – A provider informs the service registry about the web service by using it's publish ~~informs~~ interface to make the service

available to clients.
• Discover – The requestor consults the service registry to locate a published web service.
• Bind – With the information it gets from the service registry about the web service, the requestor is able to bind, or invoke, the web service.

3. What is RCP in SOAP communication model.
→ The SOAP communication model is defined by its communication style and its encoding style.
An RCP-style web service appears as a remote object to a client application. Clients express their request as a method call with a set of arguments, which returns a response containing a return value. These are represented as sets of XML elements embedded within a SOAP message.
i) RCP style supports automatic serialization/deserialization of messages, permiting developers to express a request as a method call with a set of parameters, which returns a response containing a return value.
ii) Because of this type of bilateral communication between the client and Web service.
RCP-style web services require a tightly coupled (synchronous) model of communication between the client and service provider.

* is an
• Listing 2.1 is an example of a SOAP <Body> specification. It illustrates a price quote service that requests the product price be associated with a

2022/12/10 08:15

specified plastic product. As shown fro... listing, the SOAP <Body> element contains the actual method call (as its first child element). Note the existence of the message namespace qualifier (m), which is part of the information needed for the method call to be successful. The namespace identifiers the URI of the target object. In addition, the method call requires the method name (GetProdPrice) and the parameter (prod-id).

```
<env:Envelope
    xmlns:SOAP="http://www.w3.org/2003/05/
soap-envelope"
    xmlns:m="http://www.plastics_supply.com/
prod-prices">
<env:Header>
 <tx:Transaction-id
    xmlns:t="http://www.transaction.com/transa
ctions"
    env:mustUnderstand='1'>
```

• 512

```
 </tx:Transaction-id>
 </env:Header>
 <env:Body>
 <m:GetProdPrice>
 <prod-id> 450R60P </prod-id>
 </m:GetProdPrice>
 </env:Body>
 </env:Envelops>
```

Listing 2.1 example of an RCP style of SOAP body

Once a SOAP message will containing a <Body> a call body (a method element and arguments in the

2022/12/10

<Body> element) has been sent, it is reasonable to expect that a response message will ensue. The response message will contain a <Body> element that includes the results of the remote method call. The response message corresponding to the price quote request made in Listing 2.1 could look like the response in Listing 2.2.

```
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/SOAP-
        envelope"
  xmlns:m="http://www.plastics_supply.com/
prod-prices">
<env:Header>
   <--!-optional context information-->
</env:Header>
<env:Body>
   <m:GetProdPriceResponse>
   <prod-price>134.32</prod-price>
</m:GetProdPriceResponse>
</env:Body>
<env:Envelope>
```

Listing 2.2 Example of a SOAP RCP response message.

4) What is Document (or message) in SOAP communication model.

→ SOAP can be also be used for exchanging documents containing any kind of XML data. Document-style web services are message driven. When a client invokes a message style web service, the client typically sends it an entire

document, such as a purchase order, rather than a discrete set of parameters. The web service is sent an entire document, which it processes; however, it may or may not return a response message. This style is thus asynchronous is that the client invoking the web service can continue with its computation without waiting for a response. A client that invokes a Web service does not need to wait for a response before it continues with the remainder of it's application. The response from the Web service, if any, may appear at any time later.

In the Document/literal mode of messaging:

i) A SOAP <Body> element contains an XML document fragment, a well-formed XML element that contains arbitrary application data that belongs to an XML schema and namespace separate from that of the SOAP message.

ii) The <Body> elements reflects the no explicit XML structure. The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message.

Listing 2.3 shows a purchase order SOAP message, which contains a purchase order XML document fragment ordering two injection molders of a certain type on behalf of a plastics manufacturing company. This application data upload scenario is not very well suited for an RCP-like request, such as the one shown in Listing 2.2. Instead, the application aims to

transfer the application data in one piece to a p service provider for further processing.

```
<env: Envelope
   xmlns: SOAP = "http://www.w3.org/2003/05/
soap-envelope">
<env: Header>
<tx: Transaction-id
  xmlns: t = "http://www.transaction.com/
transactions"
env: mustUnderstand$ = '1'>
512
</env: Header>
<env: Body>
<po: purchaseOrder orderDate = "2004-12-02"
   xmlns: m = "http://www.plastics-supply.com/
pos">
  <po: from>
  <po: accName> RightPlastics </po: accName>
   <po: accNumber> PSC-0343-02 </po: accNumber>
   </po: from>
   <po: to>
   <po: suppName> Plasticsupplies Inc. </po: supp
Name>
   <po: suppAddress> Yara Valley Melbourne </po:
SuppAddress>
   </po: to>
   <po: Product>
   <po: product-name> injection molder </po:
prod-name>
   <po: prod-model> G-100T </po: prod-model>
   <po: qty> 2 </po: qty>
   </po: product>
```

```
</po:PurchaseOrder>
</env:Body>
</env:Envelope>
```

listing 2.3 example of a document-style SOAP body.

5] Explain in details structure of SOAP messag with example.

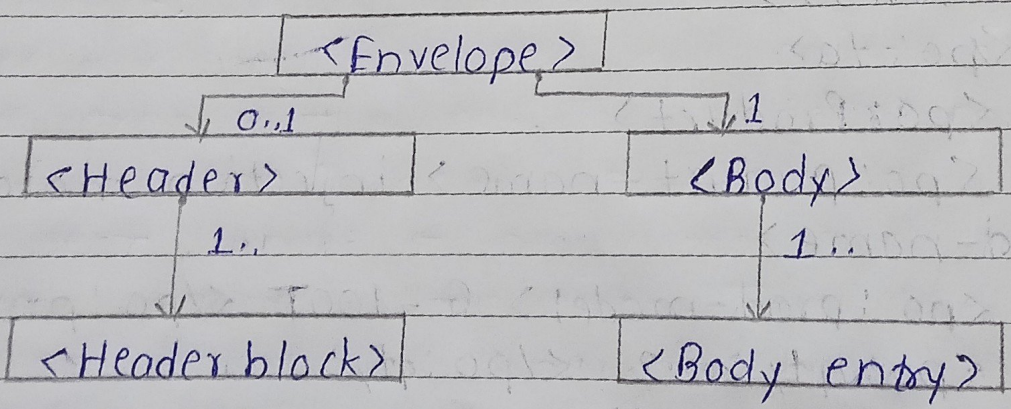
→ A SOAP message is an ordinary XML documen containing the following elements.

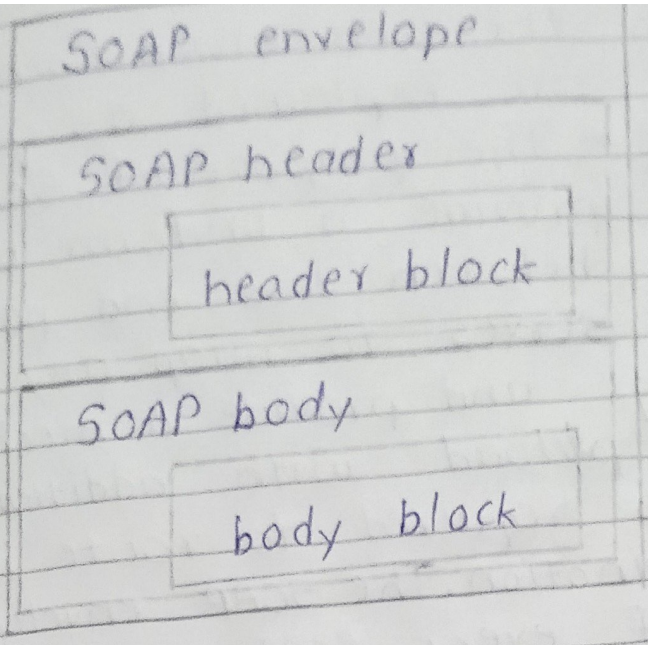
i) Envelope: Defines the start and the end of the message. It is a mandatory element.

ii) Header: Contains any optional attributes of th message used in processing the messa ge, either at an intermediary point or at the ultimate end-point. It is an optional elemen

iii) Body: Contains the XML data containing com prising the message being sent. It is a mandatory element.

iv) Fault: An optional Fault element that provides information about errors that occur while processing the message.





Structure of a SOAP message

Here is a simple sample SOAP message:

```
<?xml version="1.0"?>
<soap:Envelope
   xmlns:soap:"http://www.w3.org/2001/12/
soap-envelope">
<soap:Header>
</soap:Header>
<soap:Body>
   <--Fault element is optional,
      used only if a fault occurs in web service.
-->
  <soap:Fault>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

As we can see, a SOAP message consists of an Envelope element, inside which a Header and a Body element can be nested. Inside the Body element a Fault element can be nested, if an

error occurs in the web service. Each of these SOAP elements are explained as given below.

A) SOAP Envelope - The purpose of SOAP is to provide a uniform way to transport messages between two end points. The SOAP envelops serves to wrap any XML document interchange and provide a mechanism to augment the payload with additional information that is required to route it to its ultimate destination. The SOAP envelope is the single root of every SOAP message and must be present for the message to be SOAP compliant.

The <Envelope> element defines a framework for describing what is in a message & how to process it.

The SOAP Envelope element is the root element in a SOAP message. Inside the SOAP Envelope you find a Header (optional) and a Body element.

B) SOAP Header - SOAP divides any message into two parts contained in SOAP <Envelope>. <Header> and <Body>. The <Envelope> may contain at most the one <Header> child element. which contains all processing hints that are relevant for the end points or intermediate transport points. The <Header> may contain information about where the document shall be sent, where it originated, and may even carry digital signatures. This type of information must be separated from the SOAP <Body> which is mandatory and contains the

2022/12/10

SOAP played (the XML document).
The purpose of the <Header> is to encapsulate extensions to the message format without having to couple them to the payload or to modify the fundamental structure of SOAP.