Solution: 1) a): Some things are clearly properties of the Conceptual Class. For a book, things like the title, author, date of publication and publisher are probably conceptual. They will be independent of any physical realisation of that book. For example, there are many copies of Tolstoy's "war and peace" in existance, in many different formats, but all share the same fundamental properties.

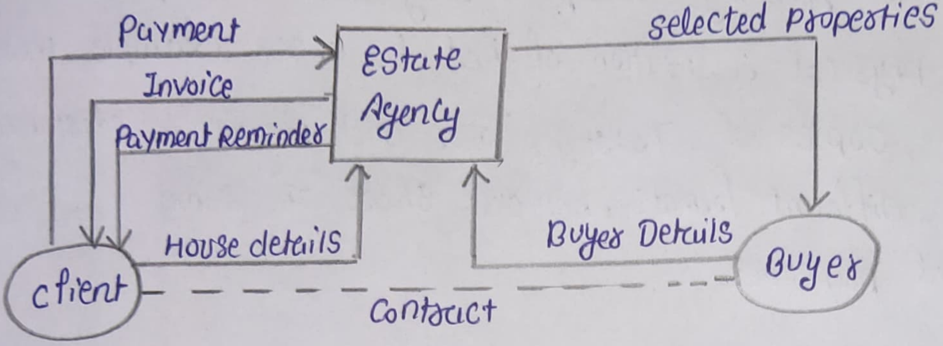
The bottom-level Classes must be Concrete, otherwise no instances will ~~b~~ ever be Created that exhibit the defined behaviour.

In a library, the shelf location and bar-code number are properties of the physical book.

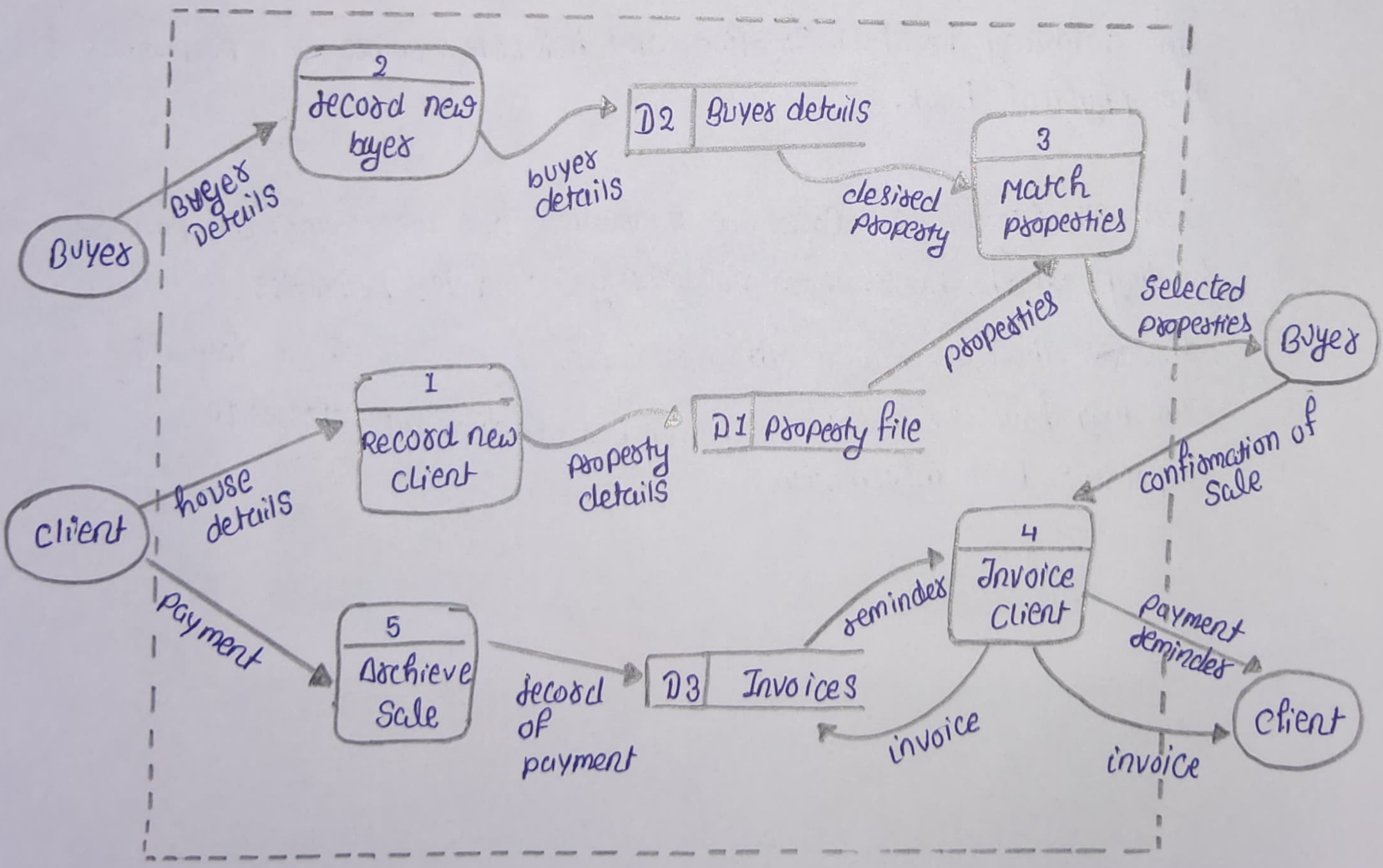
Less straightforward Cases are properties like the format (shape, size, binding) of the book. These are clearly "Concrete" properties. However there are many different Copies of each of these formats. we may have to institue a number of different classes to represent book information.

Solution :-> b)

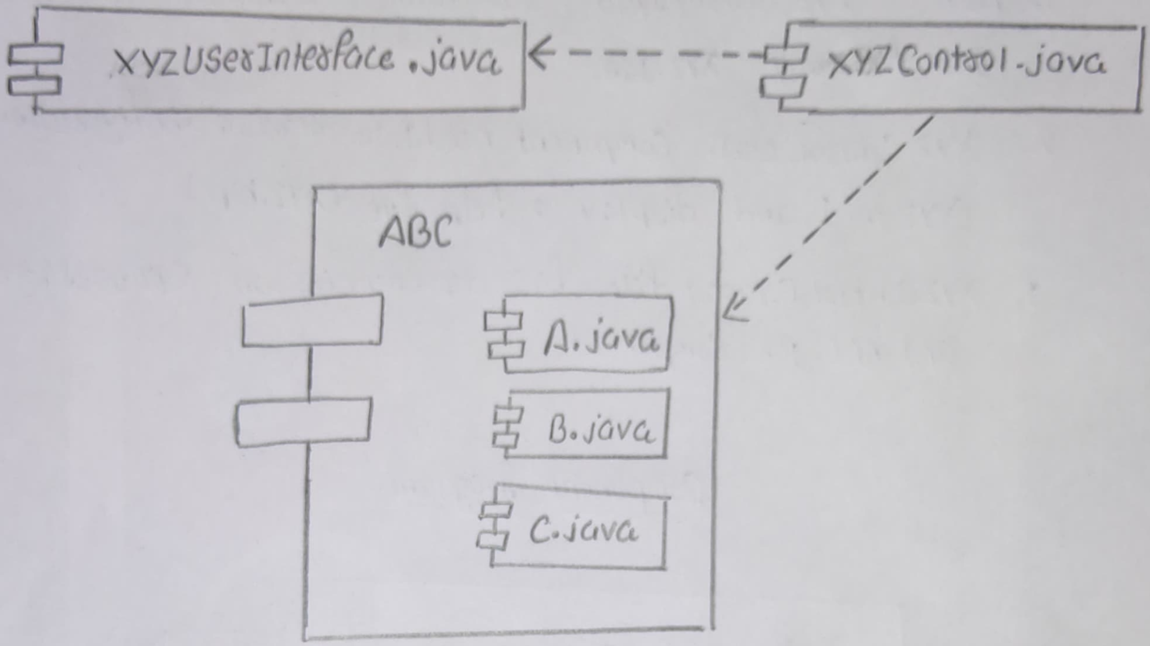
Estate Agency Data Flow Diagram (Context level):-



Estate Agency Data Flow Diagram (Level 1)



Solution 2> i>.



=> Here we have XYZ UserInterface.java and XYZControl.java are representing two classes XYZ UserInterface and XYZ Control respectively.

=> ABC Components has 3 classes A, B and C and each of them are implemented by (.java) file.

=> XYZControl has a dependency on the ABC Component and XYZUserInterface Component.

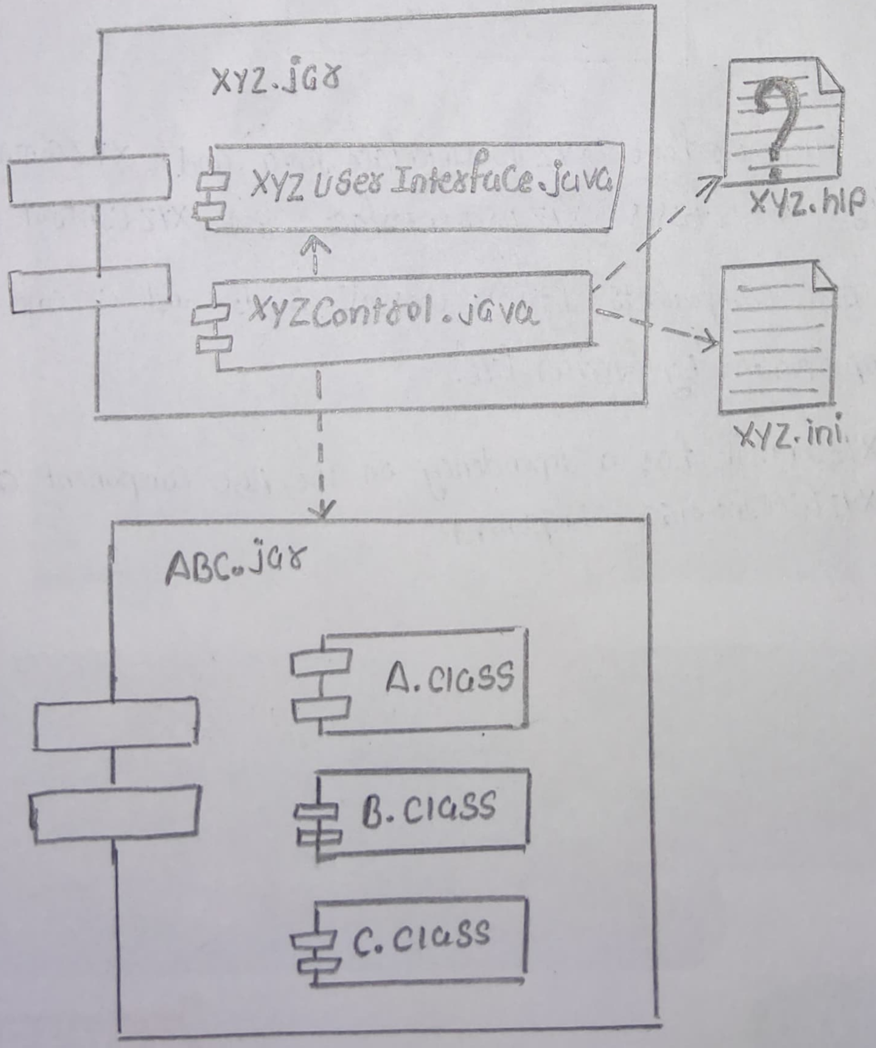
Solution 2> ii>

1. both XYZUserInterface class and XYZ Control are grouped into jar file say XYZ.jar.

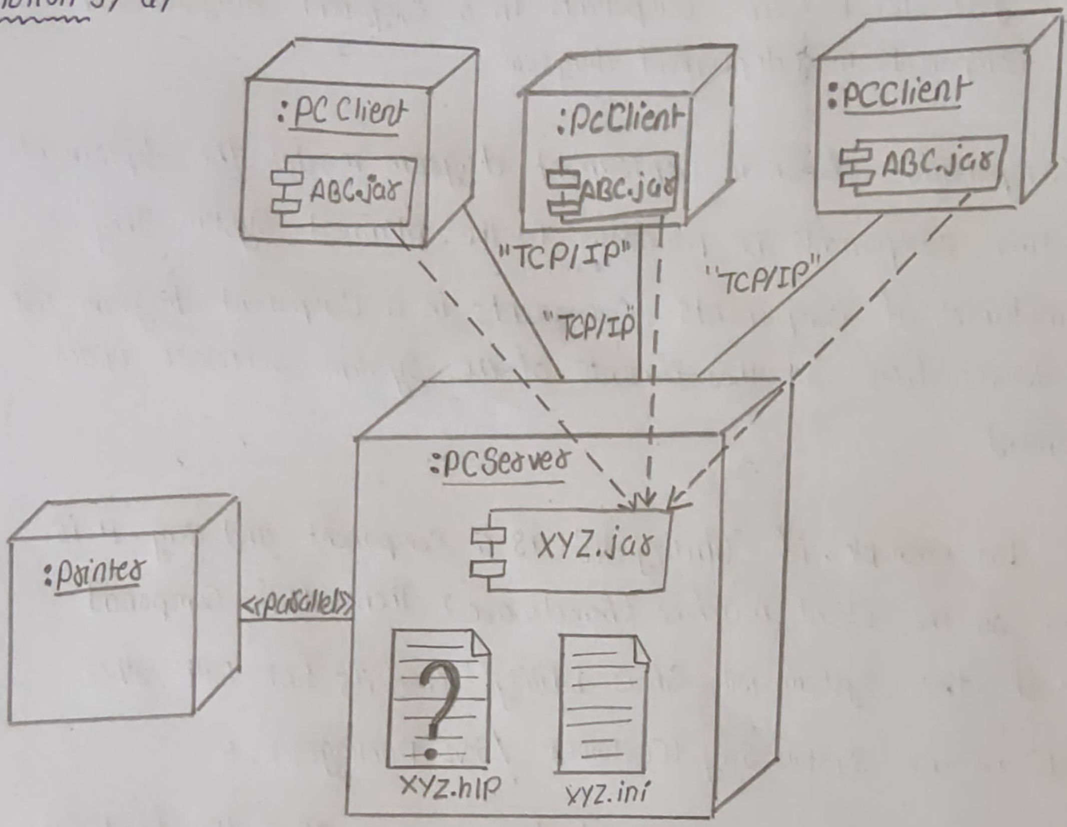
2. XYZ Control.class Component needs to read a Configuration file (XYZ.ini) and display a help file. (XYZ.hlp)

3. XYZ Control (.java) file has dependencies on XYZ userinterface.java and ABC.jar Component

Component Diagram:



Solution 3) a)



⇒ Deployment diagram

⇒ 3 nodes are three client PCs, a server PC and a printer.

⇒ Client PC contains ABC.jar Component

⇒ Server PC contains XYZ.jar Component with Configuration files XYZ.ini and XYZ.hlp.

⇒ also ABC.jar component has dependencies on the XYZ.jar Components

⇒ Communications between clients and server are following TCP/IP protocols.

⇒ Printer is parallely connected to server.

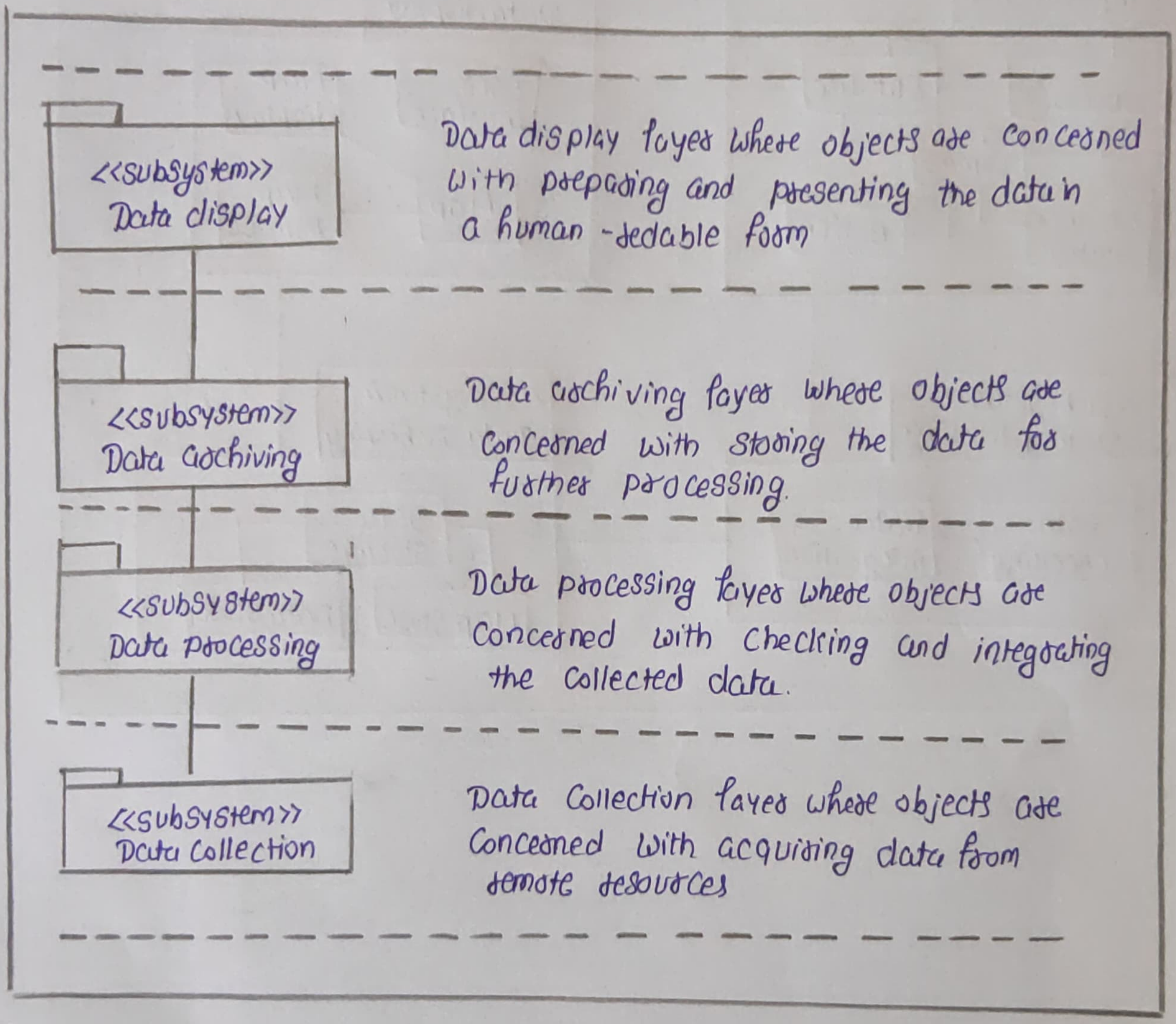
Solution: 3) b) Difference between Components in a Component diagram and Components in a deployment diagram.

Components shown in deployment diagram model the deployment of run-time components on processor in the planned System. They are normally instances of Components. Components in a Component diagram are usually Source files or other elements of the System Software under development.

For example, if 'utility.dll' is a Component and say it is deployed on the client machine (hardware). Then, the Component diagram of this system will show utility and its link with other Components in the System say (Customer /SQL packages).

whereas, the Deployment diagram will show the hardware Configuration - DB server / web server / Client Machine and utility Component will be placed into Client machine.

Solution: 4) i) Layered architecture for above weather mapping system.



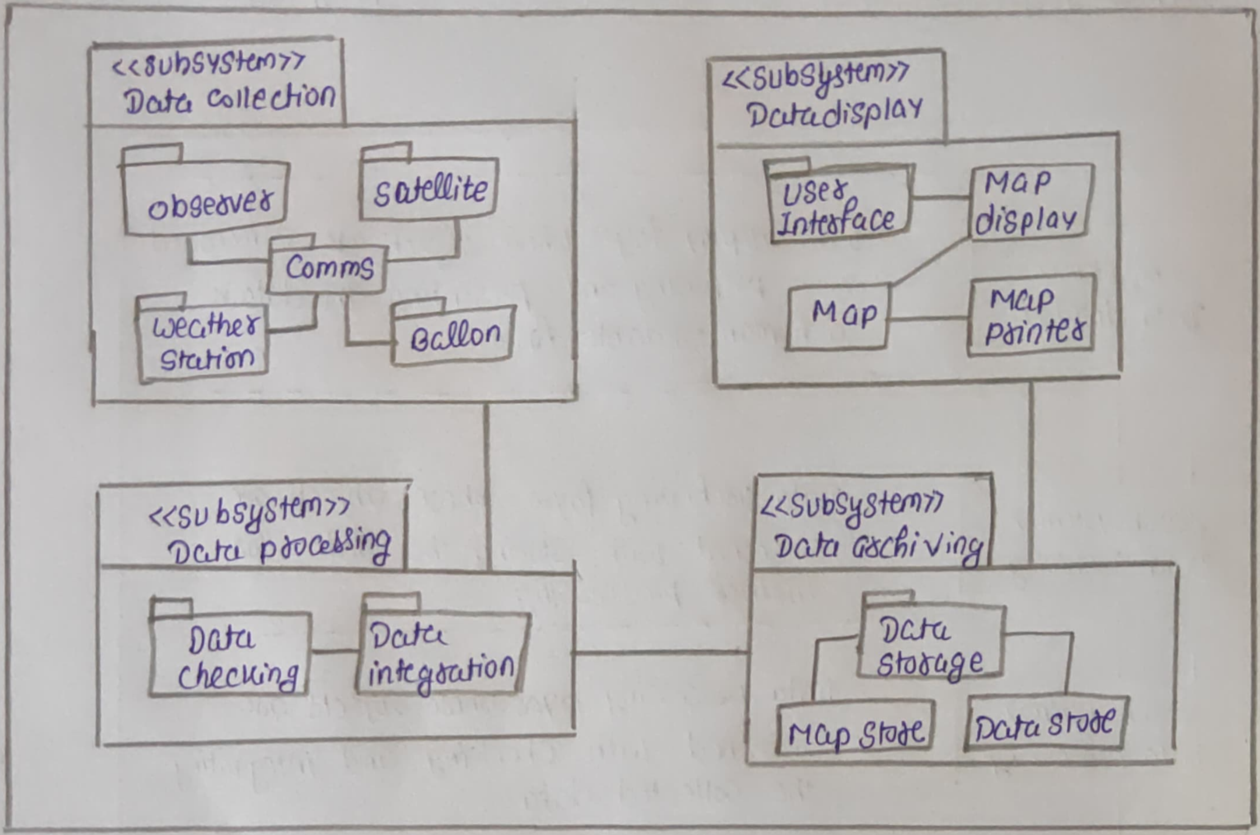
Data display layer where objects are concerned with preparing and presenting the data in a human-readable form

Data archiving layer where objects are concerned with storing the data for further processing.

Data processing layer where objects are concerned with checking and integrating the collected data.

Data Collection layer where objects are concerned with acquiring data from remote resources

Solution 4> ii>


<<subsystem>>
Data collection
observer
Satellite
Comms
weather Station
Ballon
<<subsystem>>
Datadisplay
User Interface
Map display
Map
Map Printer
<<subsystem>>
Data processing
Data Checking
Data integration
<<Subsystem>>
Data archiving
Data Storage
Map Store
Data Store