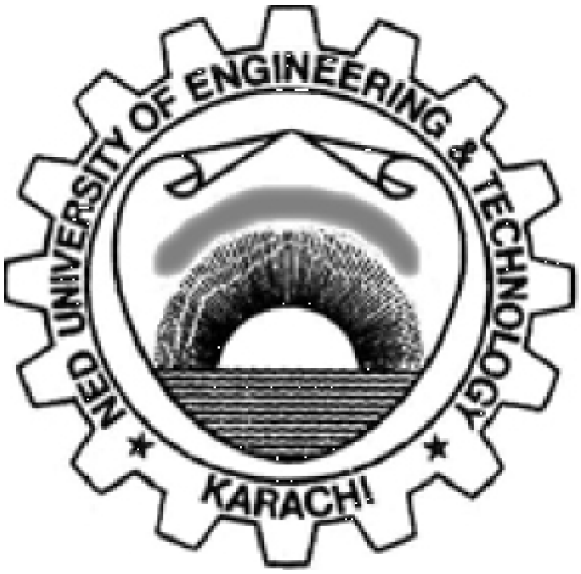
Workbook

Software Design & Architecture
(SE-308)



Name: Kabeer Ahmed
Roll no: SE-19028
Batch: 2019
Year: 2021
Department: Software Engineering

Kabeer Ahmed
SE-19028

Workbook

Software Design & Architecture
(SE – 308)

Prepared by

Dr. Shehnila Zardari
Assistant Professor, Department of Software Engineering

Approved by Chairman
Department of Software Engineering

Kabeer Ahmed
SE-19028

TABLE OF CONTENT

Kabeer Ahmed
SE-19028

LAB # 1

Object:
Introduction to Unified Modeling Language (UML)

Theory:
The Unified Modeling Language (UML) enables system builders to create blueprints that capture their visions in a standard, easy-to understand way, and provides a mechanism to effectively share and communicate these visions with others. It does this via a set of symbols and diagrams. Each diagram plays a different role within the development process.

Components of the UML

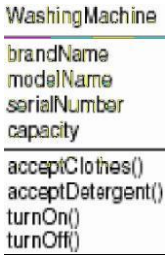
- The UML consists of a number of graphical elements that combine to form diagrams.
- UML is a language; it has rules for combining these elements.
- The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a **model.**
- A UML model tells what a system is supposed to do.

UML Diagrams

- Class Diagram
- Object Diagram
- Use Case Diagram
- State Diagram
- Sequence Diagram Activity Diagram
- Communication Diagram
- Component Diagram • Deployment Diagram

Class Diagram:

A class is a category or group of things that have the same attributes and the same behaviors. A rectangle is the icon that represents the class. It's divided into three areas. The uppermost area contains the name, the middle area holds the attributes, and the lowest area holds the operations i.e. anything in the class of washing machines has attributes such as brand name, model, serial number, and capacity. Behaviors for things in this class include the operations "accept clothes," "accept detergent," "turn on," and "turn off.



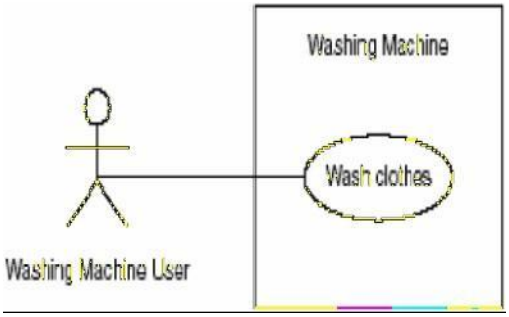
In UML, a multiword class name has initial capital letters for all the words and eliminates white space between each word (for example, Washing Machine). Attribute names and operation names follow the same convention, but the first letter of the first word isn't capitalized (for example, acceptClothes
()).

Object Diagram:

An object is an instance of a class—a specific thing that has specific values of the class's attributes. The icon is a rectangle, just like the class icon, but the name is underlined. In the icon on the left, the name of the specific instance is on the left side of a colon, and the name of the class is on the right side of the colon. The name of the instance begins with a lowercase letter. It's also possible to have an anonymous object, as the icon on the right of Figure shows. This just means that you don't supply a specific name for the object, although you do show the class it belongs to.

Use Case Diagram:

A use case is a description of a system's behavior from a user's standpoint. For system developers, the use case is a valuable tool: It's a tried-and-true technique for gathering system requirements from a user's point of view. Obtaining information from the user's point of view is important if the goal is to build a system that real people can use.



Kabeer Ahmed
SE-19028

Lab#01

Exercise:

1. List the advantages of Unified Modeling Language.

ADVANTAGES OF UML

- A UML diagram is a visual representation of the relationships between classes and entities in a computer program.
- A UML diagram is beneficial in that it is very readable. The diagram is meant to be understood by any type of programmer and helps to explain relationships in a program in a straightforward manner.
- UML is the current standard for programming in object-oriented programming languages. When creating classes and other objects with relationships between each other, UML is what is used to visually describe these relationships.
- UML helps to plan a program before the programming takes place. In some tools used to model UML, the tool will generate code based on the classes set up in the model.
- UML is a highly recognized and understood platform for software design. It is a standard notation among software developers.
- The software architecture is the blueprint of the system. It is the framework on which the efficiency of the system and its processes depend.
- UML is a rich and extensive language that can be used to model not just object-oriented software engineering, but application structure and behavior, and business processes too.

Kabeer Ahmed
SE-19028

Lab # 2

Object:
Introduction to Rational Rose

Theory:
Once the Requirements have been gathered, it is necessary to convert them into an appropriate design. In order to bridge the requirements gathering and design phases, we use some modeling or specification tool, e.g. the Unified Modeling Language. A commonly available CASE tool for design through UML is the Rational Rose. Given below is an overview of the various aspects in which development could be done using Rational Rose.

The Browser:
The Browser contains a list of all the modeling elements in the current model. The Browser contains a tree view of all the elements in the current Rose model. Elements in the tree structure are either compressed or expanded.

The Documentation Window:
The Documentation Window may be used to create, review, and modify for any selected modeling element. If nothing is selected, the window displays the string "No selection". The contents of the Documentation Window will change as different modeling elements are selected.

To create the documentation for a selected element, click to select the element and enter its documentation in the Documentation Window.

The Documentation Window may be docked or floating just like the Browser. Visibility of the window is controlled via the View: Documentation window command.

Diagram Windows:
Diagram windows allow you to create and modify graphical views of the current model. Each icon on a diagram represents a modeling element. Since diagrams are used to illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams.

Views in Rational Rose:
There are many views of a software project under development. Rational Rose is organized around the views of a software project (4 + 1 View):

- The Use Case View.
- The Logical View.
- The Component View.
- The Deployment View.

Each view presents a different aspect of the visual model. Each view contains a Main diagram by default. Additional elements and diagrams are added to each view throughout the analysis and design process

The Use Case View:

The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

Packages in the Use Case View:

Packages in the use case view can contain actors, use cases, sequence diagrams, and/or collaboration diagrams. To create a package:

1. Right-click on the parent modeling element (use case view or another package) to make the shortcut menu visible.
2. Select the New: Package menu command. This will add a new package called NewPackage to the browser.
3. While the new package is still selected, enter its name.

Once a package is created, modeling elements may be moved to the package. To move a modeling element to a package:

- Click to select the modeling element to be relocated.
- Drag the modeling element to the package.

The Logical View:

The logical view of the system addresses the functional requirements of the system. This view looks at classes and their static relationships. It also addresses the dynamic nature of the classes and their interactions. The diagrams in this view are class diagrams and state transition diagrams.

Class Diagram:

The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by doubleclicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.

Kabeer Ahmed
SE-19028

LAB#02

Exercise:

1. What is 4 + 1 view? Briefly define each view

4+1 view:

4+1 is a view model designed by Philippe Kruchten for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineer, and project managers. The four views of the model are logical, development, process and physical view. In addition selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view. Hence the model contains 4+1 views:

- **Logical view:** The logical view is concerned with the functionality that the system provides to end-users. UML diagrams are used to represent the logical view, and include class diagrams, and state diagrams.
- **Process view:** The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the run time behavior of the system. The process view addresses concurrency, distribution, integrator, performance, and scalability, etc. UML diagrams to represent process view include the activity diagram.
- **Development view:** The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the Package diagram.
- **Physical view:** The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. UML diagrams used to represent the physical view include the deployment diagram.
- **Scenarios:** The description of an architecture is illustrated using a small set of use cases, or scenarios, which become a fifth view. The scenarios describe sequences of interactions between objects and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. This view is also known as the **use case view.**

The 4+1 view model is generic and is not restricted to any notation, tool or design method. Quoting Kruchten,

The “4+1” view model is rather “generic”: other notations and tools can be used, other design methods can be used, especially for the logical and process decompositions, but we have indicated the ones we have used with success.

Kabeer Ahmed
SE-19028

LAB # 3

Object:
To understand Use Case View

Theory:
The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

Actor
An actor is represented by a stickman. To create an actor:

1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Actor menu command. This will add an actor called NewClass to the browser.
3. While the new class is still selected, enter the name of the actor.

As actors are created, their documentation is added to the model. To add the documentation for an actor:

1. Click to select the actor in the browser.
2. Position the cursor in the Documentation Window.
3. Enter the documentation for the actor.

Each modeling element is associated with a Specification window. The Specification contains additional information about the element. To view the Specification for an actor:

1. Right-click to select the actor in the browser and make the shortcut menu visible.
2. Select the Specification menu command.

When you view the Specification for an actor, you will notice that the title is "Class Specification for <actor>". This is due to the fact that an actor is a class in Rose with a stereotype /*Extension of Metaclass*/ of Actor.

Use Case.
A use case is represented by an oval. To create a use case:

1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Use Case menu command. This will add an unnamed use case to the browser.
3. While the new use case is still selected, enter the name of the use case.

Use cases are documented in two ways – they have a brief description and a flow of events. To add the brief description for a use case:

1. Click to select the use case in the browser.

2. Position the cursor in the Documentation Window.
3. Enter the brief description for the use case.

The flow of events is captured in documents and/or web pages external to Rose. The documents and web pointers are linked to the use case. To link an external document or web pointer to the use case:

1. Right-click to select the use case in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Files tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert File menu command and enter the name of the document to be linked to the use case.
6. Or, select the Insert URL menu command and enter the www pointer to be linked to the use case.
7. Click the Open button.

Use Case Diagrams.

A use case diagram shows the relationships among actors and use cases within the system. The use case view is automatically created with one use case diagram called Main. A system may have other use case diagrams (e.g. a diagram to show all the use cases for one actor).

To open a use case diagram:

1. Click the + next to the Use Case View in the browser to expand the view.
2. Double-click on the diagram you wish to open.
3. Re-size the diagram as needed.

To add an actor or use case to the diagram:

1. Click to select the actor or use case in the browser.
2. Drag the actor or use case to the diagram.

The main type of relationship is a communication relationship. This relationship is shown as a unidirectional association. To add communication relationships to the diagram:

1. Click to select the uni-directional icon from the toolbar.
2. Click on the actor or use case initiating the communication.
3. Drag the relationship line to the participating use case or actor.

A use case diagram may have two other types of relationships – extends and uses relationships. Extends and uses relationships are shown as generalization relationships with stereotypes. To create an extends or uses relationship:

Kabeer Ahmed
SE-19028

LAB#03

Exercise:

1. Draw a Use-Case diagram for an Online Shopping System

2. Communication diagram of an ATM machine. How user interact with the machine and Draw how objects collaborate with each other to complete transaction.

Kabeer Ahmed
SE-19028

LAB # 4

Object:
To understand Class Diagram

Theory:
The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by doubleclicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.

Each package typically has its own main diagram which is a picture of its key packages and classes. To create the Main class diagram for a package, double-click on the package on a class diagram. Once the Main diagram is created for a package, you can add packages and classes to the diagram by selecting them in the browser and dragging them onto the diagram.

Classes from any package may be added to a class diagram by selecting the class on the browser and dragging it onto the open class diagram. If the Show Visibility option is set to true either as a default using the Tool: Options menu or individually by using the shortcut menu for the class, the name of the "owning" package is displayed. The package name will be visible for all classes that do not belong to the package owning the class diagram.

Packages and classes may also be created using the class diagram toolbar. To create a package or class using the toolbar:

1. Click to select the icon (package or class) on the class diagram toolbar.
2. Click on the class diagram to place the package or class.
3. While the new package or class is still selected, enter its name.
4. Multiple packages or classes may be created by depressing and holding the Shift key.

Packages and classes created on class diagrams are automatically added to the browser.

To create a class diagram:

1. Right-click to select the owning package and make the shortcut menu visible.
2. Select the New: Class Diagram menu command. This will add a class diagram called NewDiagram to the browser.
3. While the new class diagram is still selected, enter its name.
4. To open the class diagram, double-click on it in the browser.

Class Structure

The structure of a class is represented by its set of attributes. Attributes may be created in the browser, via the Class Specification or on a class diagram.

To create an attribute in the browser:

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. To create an attribute, select the New: Attribute menu command. This will add an attribute called name to the browser.
3. While the new attribute is still selected, enter its name.
4. Attribute data types and default values may not be entered via the browser.

To create an attribute using the Class Specification:

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Attributes tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an attribute called name.
6. While the new attribute is still selected, enter its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

To create an attribute on a class diagram:

1. Right-click to select the class on the class diagram and make the shortcut menu visible.
2. Select the Insert New Attribute menu command. This will insert an attribute in the form. name : type = initval
3. While the attribute is still selected, fill in its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

Attributes of a class may be viewed in the browser. The class will be initially collapsed. To expand the class to view its attributes, click the + next to the class.

Attributes should be documented. **To add the documentation for an attribute:**

1. Click to select the attribute in the browser.
2. Position the cursor in the Documentation Window. If the Documentation Window is not visible, select the View: Documentation menu command.
3. Enter the documentation for the attribute.

To delete an attribute:

1. Right-click to select the attribute in the browser or on the Attributes tab of the Class Specification and make the shortcut menu visible.
2. Select the Delete menu command.

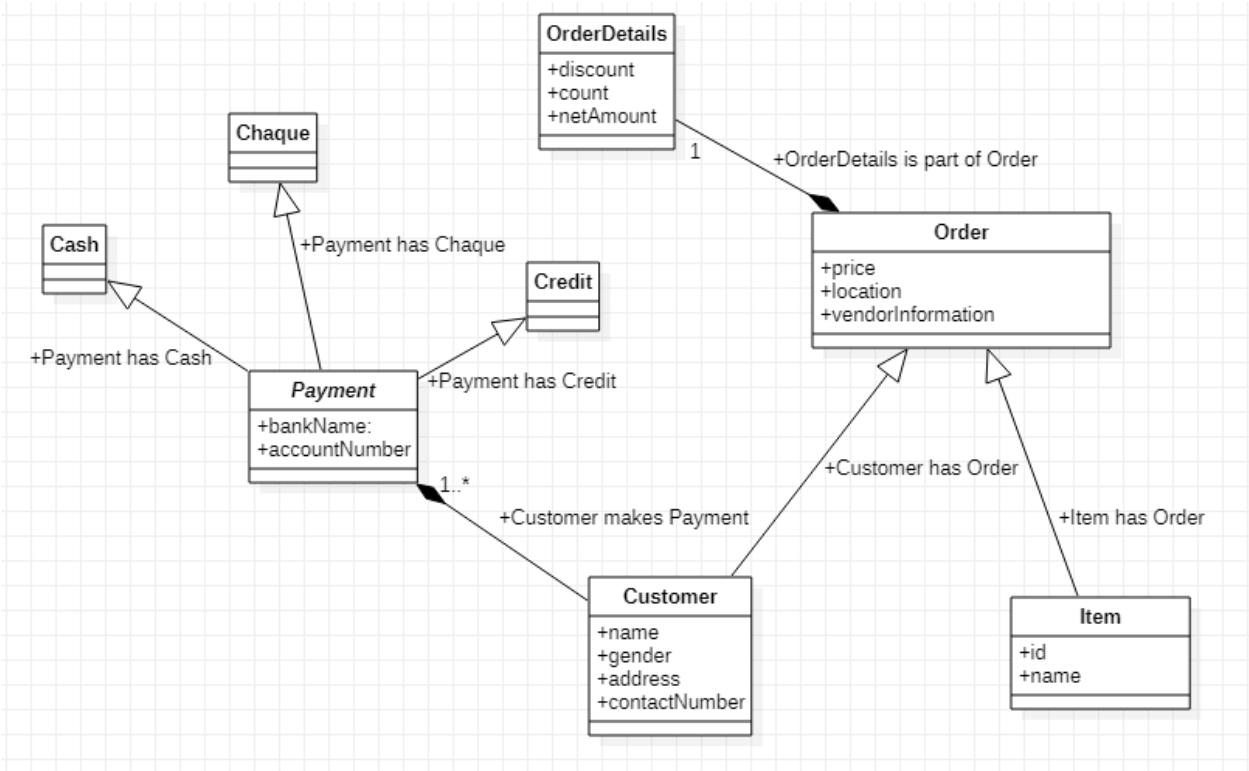
Kabeer Ahmed
SE-19028

LAB#04

Exercise:

1. **Draw a class diagram for Retail Catalog Order**

ANS:



Kabeer Ahmed
SE-19028

LAB # 5

Object:

To understand State Transition Diagram.

Theory:

State transition diagrams show the life history of a given class, the events that cause a transition from a state, and the actions that result from a state change. They are created for classes whose objects exhibit significant dynamic behavior.

To create a state transition diagram:

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the State Diagram menu command.

To open a state transition diagram.

1. Click the + next to the class to expand the tree
2. Double-click on the State Diagram for the class

States.

A state is represented by an oval.

To create a state.

1. Click to select the State icon from the toolbar.
2. Click on the state transition diagram to place the state.
3. While the state is still selected, enter its name.

State Transitions.

A state transition is represented as an arrow which points from the originating state to the successor state.

To create a state transition.

1. Click to select the state transition arrow from the toolbar.
2. Click on the originating state and drag the arrow to the successor state.

State Actions.

Behavior that occurs while an object is in a state can be expressed in three ways: entry actions , activities , and exit actions . The behavior may be a simple event or it may be an event sent to another object.

To create an entry action, exit action or activity.

1. Point to the state and double click to make the State Specification dialog box visible.
2. Select the Detail tab.
3. Click-right in the Actions field to make the pop-up menu visible.
4. Select the Insert menu choice to insert a new action called entry.
5. Double-click on the action to make the State Action Specification visible.
6. If the action is a simple action, enter the name of the action in the Action field.
7. If the action is a send event action, enter the name of the event to be sent in the Send Event field. If the event has arguments, enter the arguments in the Send Arguments field. Enter the name of the

target object (object receiving the event) in the Send Target field.

8. Select the appropriate radio button in the When field (On Entry to create an entry action, On Exit to create an exit action, or Entry Until Exit to create an activity.
9. Click the OK button to close the Action Specification.
10. Click the OK button to close the State Specification.

Start and Stop States.

There are two special states associated with state transition diagrams – the start state and the stop state. The start state is represented by a filled in circle and the stop state represented by a bull's eye.

To create a start state.

1. Click to select the start state icon from the toolbar.
2. Click on the diagram to place the start state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the start state and drag the state transition arrow to the successor state.

To create a stop state.

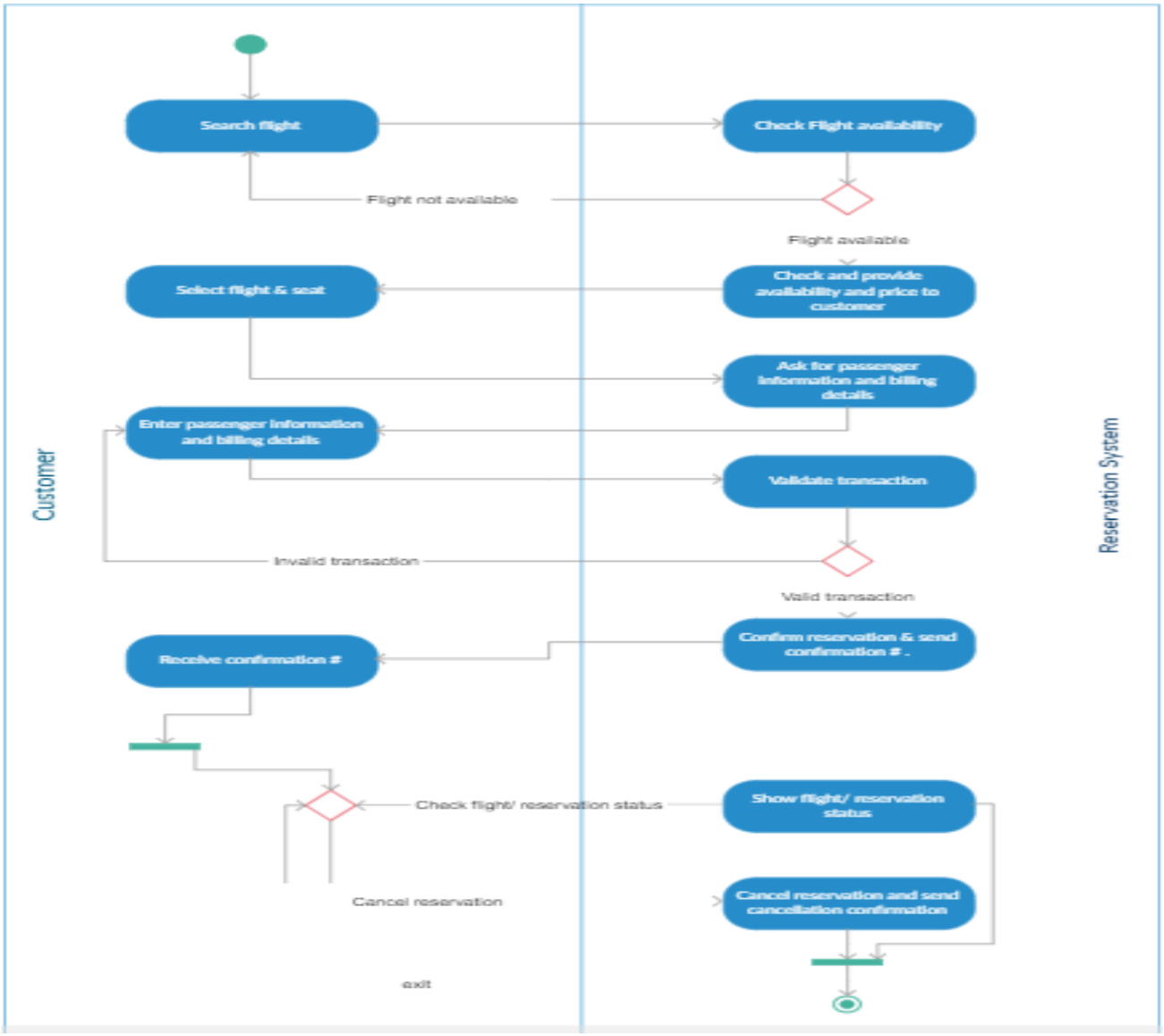
1. Click to select the stop state icon from the toolbar.
2. Click on the diagram to place the stop state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the originating state and drag the state transition arrow to the stop state.

Kabeer Ahmed
SE-19028

LAB#05

Exercise:

1. Draw State and Activity Diagram for Airline Reservation System. Show different activities involved for making Reservation, Changing Reservation and Passenger Checking in for flight.



Kabeer Ahmed
SE-19028

LAB # 6

Object:
To understand System Modeling.

Theory:
Modeling consists of building an abstraction of reality. These abstractions are simplifications because they ignore irrelevant details and they only represent the relevant details (what is relevant or irrelevant depends on the purpose of the model).

Why Model Software?
Software is getting larger, not smaller; for example, Windows XP has more than 40 million lines of code. A single programmer cannot manage this amount of code in its entirety. Code is often not directly understandable by developers who did not participate in the development; thus, we need simpler representations for complex systems (modeling is a mean for dealing with complexity).
A wide variety of models have been in use within various engineering disciplines for a long time. In software engineering a number of modeling methods are also available.

Analysis Model Objectives.

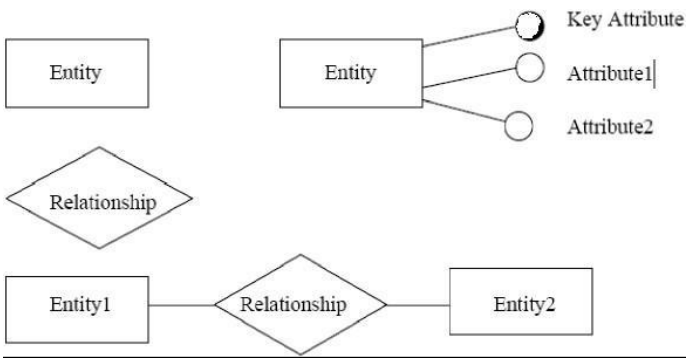
- To describe what the customer requires.
- To establish a basis for the creation of a software design.
- To define a set of requirements that can be validated once the software is built.

The Elements of the Analysis Model.
The generic analysis model consists of:

- An entity-relationship diagram (data model).
- A data flow diagram (functional model).

Entity Relationship Diagram.
An entity relationship diagram (ERD) is one means of representing the objects and their relationships in the data model for a software product.

Entity Relationship diagram notation.



- **Creating an ERD**

Here are the steps you may follow to create an entity-relationship diagram.

- **Identify Entities**

Identify the entities. These are typically the nouns and noun-phrases in the descriptive data produced in your analysis. Do not include entities that are irrelevant to your domain.

- **Find Relationships**

Discover the semantic relationships between entities. These are usually the verbs that connect the nouns. Not all relationships are this blatant; you may have to discover some on your own. The easiest way to see all possible relationships is to build a table with the entities across the columns and down the rows, and fill in those cells where a relationship exists between entities.

- **Draw Rough ERD**

Draw the entities and relationships that you have discovered.

- **Fill in Cardinality**

Determine the cardinality of the relationships. You may want to decide on cardinality when you are creating a relationship table.

- **Define Primary Keys.**

Identify attribute(s) that uniquely identify each occurrence of that entity.

- **Draw Key-Based ERD.**

Now add them (the primary key attributes) to your ERD. Revise your diagram to eliminate many-tomany relationships, and tag all foreign keys.

- **Identify Attributes.**

Identify all entity characteristics relevant to the domain being analyzed.

- **Map Attributes.**

Determine which to entity each characteristic belongs. Do not duplicate attributes across entities. If necessary, contain them in a new, related, entity.

- **Draw fully attributed ERD.**

Now add these attributes. The diagram may need to be modified to accommodate necessary new entities.

- **Check Results.**

Is the diagram a consistent and complete representation of the domain.

Data Flow Diagram

A **data flow diagram (DFD)** is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

Kabeer Ahmed
SE-19028

LAB#06

Exercise:

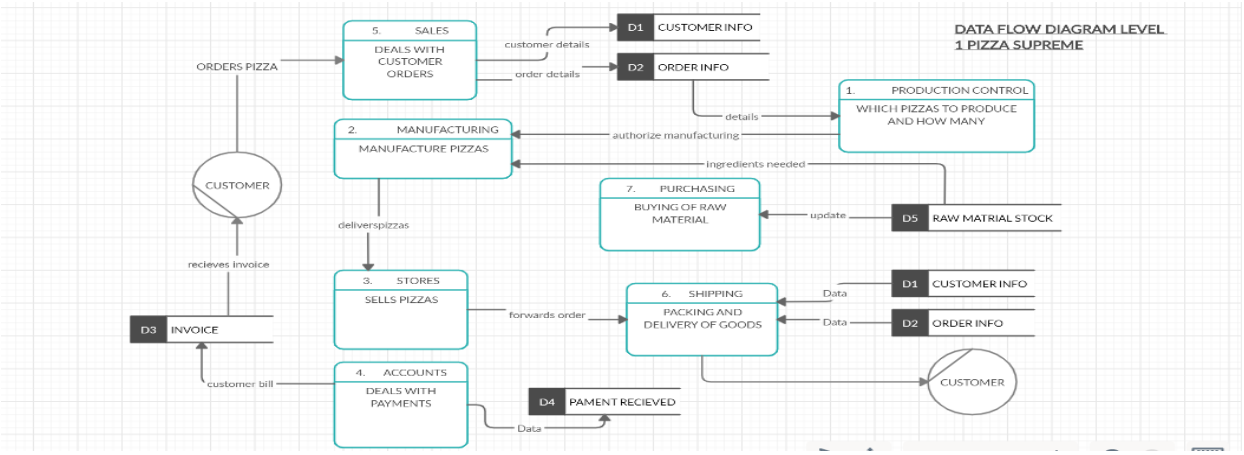
1**. Draw context diagram and Level 1 diagram of the following case study**

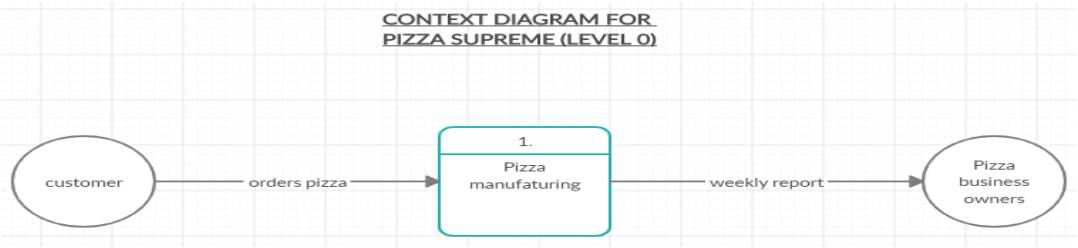
Case Study – Pizza Supreme

A large pizza business makes pizzas and sells them. The pizzas are manufactured and kept in cold storage for not more than two weeks.

The business is split into a number of functional units. There is Production Control, Manufacturing, Stores, Accounts, Sales, Shipping and Purchasing. Production Control are responsible for organizing which pizzas to produce in what order and in what quantity. They need to schedule the production of the pizzas according to the current and expected sales orders together with the number of pizzas already in Stores. Manufacturing take the raw materials from the Stores and manufacture pizzas returning the completed goods to the Stores. Accounts deal with the payments for the pizzas when delivered to the customer and the payment to the suppliers of the raw materials.

When a sales order is received by sales they record what is being ordered and by whom. They also record the details of the expected date of delivery. Production Control access this information and make sure that, if required, pizzas are produced by Manufacturing and are ready in Stores for when the delivery needs to be made. After the delivery is made Accounts make sure that the customer receives an invoice and that payment for the invoice is received at which time a receipt is issued. Purchasing look at the current stock of raw materials and by using current stock levels, supplier turnaround times and quantity to be ordered decide what needs to be ordered on a daily basis. Their aim is never to run out of an ingredient but to minimize the amount of raw material kept in stock





Kabeer Ahmed
SE-19028

LAB # 7

Object:
User Interface Prototyping using GUI Design Studio.

Theory:
User interface (UI) prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. UI prototypes have several purposes:

- As an analysis artifact that enables you to explore the problem space with your stakeholders.
- As a requirements artifact to initially envision the system.
- As a design artifact that enables you to explore the solution space of your system.
- A vehicle for you to communicate the possible UI design(s) of your system.
- A potential foundation from which to continue developing the system (if you intend to throw the prototype away and start over from scratch then you don't need to invest the time writing quality code for your prototype).

GUI Design Studio
GUI Design Studio is a code-free, drag and drop user interface design and prototyping tool for creators of Web, Desktop, Mobile and Embedded software applications.

- Design screens or web pages • Add user interaction behavior. • Test as a running prototype

Software Designers, User Experience Professionals, Business Analysts, Developers, Project Managers and Consultants can use it to. • Document product ideas

- Create project proposals
- Create design mock-ups

We can use it to quickly lay down ideas and test them out in a low-cost, risk-free environment and avoid expensive implementation rework by getting the right design agreed upon first.

- Verify designs and requirements
- Explore alternatives
- Evaluate different usage scenarios

The Project panel on the Design Bar provides access to all of your project design documents and image files. From here you can quickly create new project folders to organize your files, create new design documents, duplicate existing designs and import images from the clipboard.

The Elements panel on the Design Bar provides access to all of the windows and controls that can be used to create an application GUI. Elements are organized into categories. When you select a category from the list, its elements appear in the palette window below.

The Icons panel on the Design Bar provides access to common icon images and those within the main project and any others that have been linked in from the Project panel. The common icons are organized into categories. You can create additional category folders for new icons.

The Annotations panel on the Design Bar provides access to special elements such as overlay text boxes, highlight rings and markers. The annotation elements always appear on top of other design elements.

The Storyboard panel on the Design Bar provides access to special elements for building flow control in a design to create a working prototype. The storyboard elements always appear on top of other design elements.

The Data panel on the Design Bar provides access to the Data Tables within your project.

The Notes panel on the Design Bar provides editors for recording notes associated with a design document and its elements. These can be used to provide popup descriptions and to generate specification documents. Unlike all of the other panels, the content of the Notes panel changes to reflect the active design document.

By default, the working area background for designs is plain white. You can change this in the Preference settings.

With GUI Design Studio, all of your work will normally be done within **projects.** A project has its own folder structure on disk and may contain design documents, bitmap image files and project-specific icons. The project file is stored within the project folder and has the .GDP extension. Sometimes you will want to access the files within one project from another project, such as:

- Using a library of common design components.
- Referencing a set of GUI design guidelines.
- Building on an older version of a project within a new development project
- Using another project as a reference example.

Creating a new project

Menu Command: File then New Project

When you create a new project you must enter a name and select a root folder for the project files.

Creating a new project

Menu Command: File then New Project then New Design

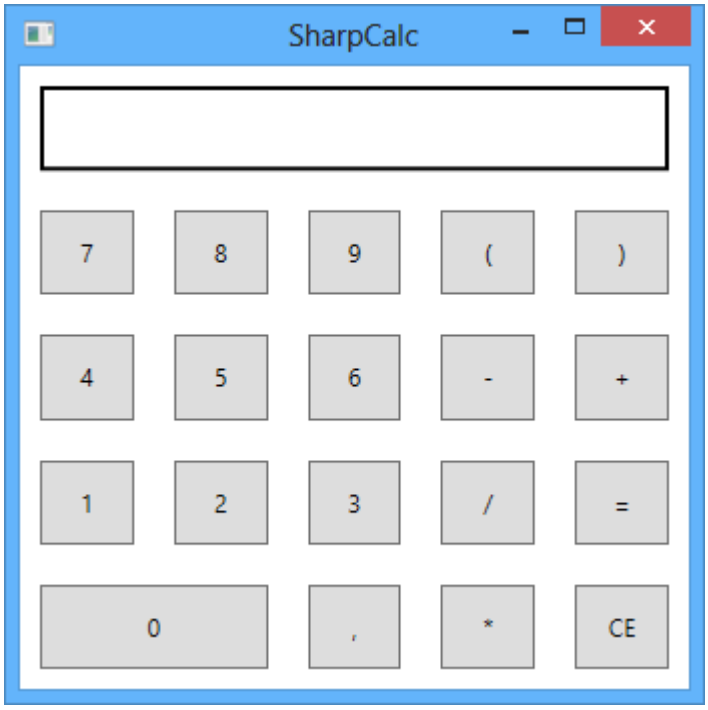
There are two methods for creating design documents. The first uses the File | New menu command or “New” toolbar button to create and open a blank, untitled document ready for editing.

Kabeer Ahmed
SE-19028

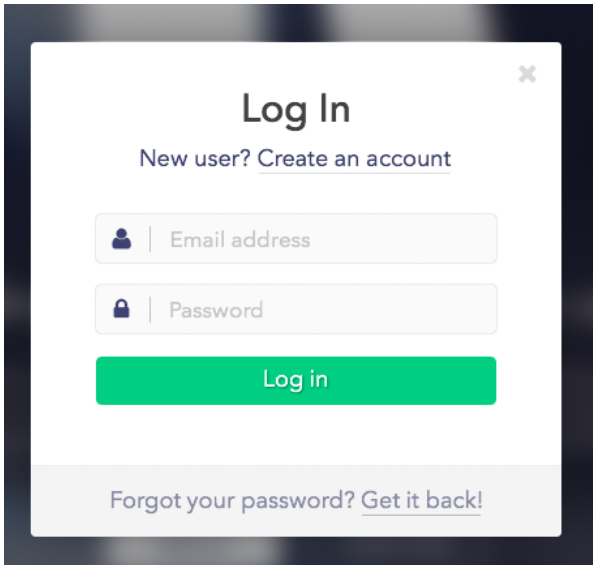
LAB#07

Exercise:

1. Make an interface for a simple calculator.



Make an interface for a login window.

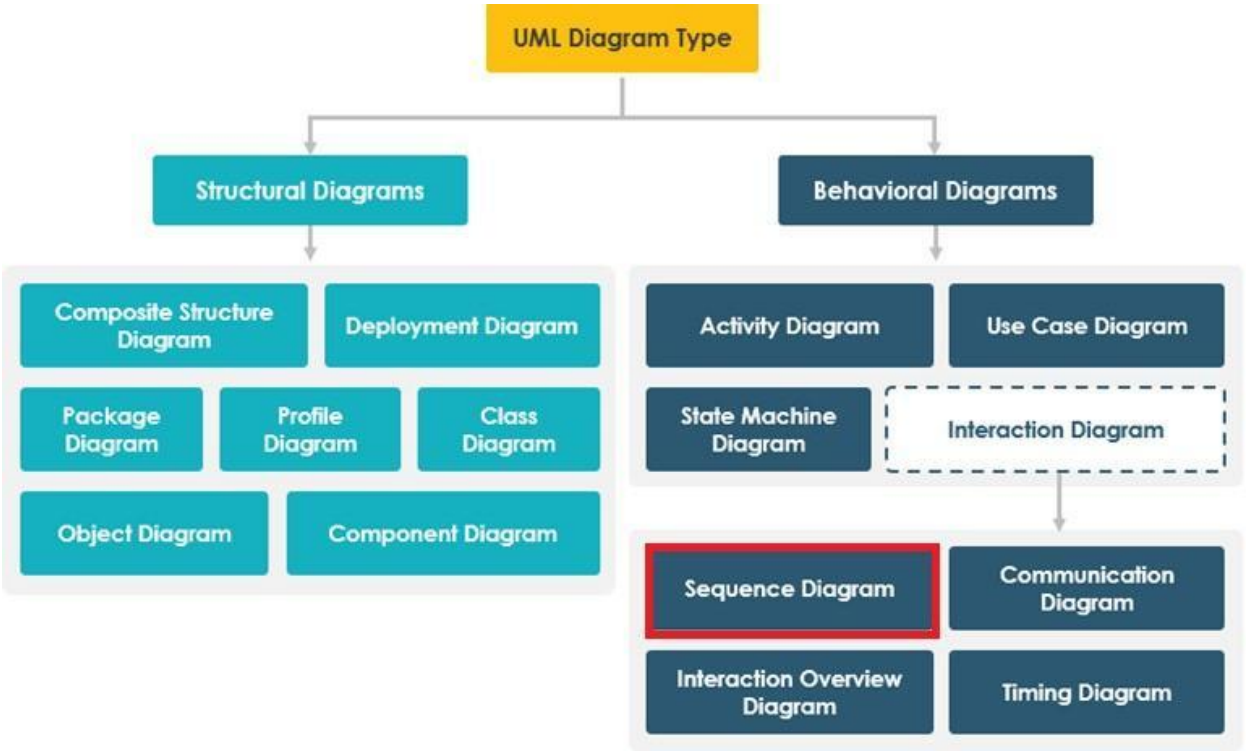


Kabeer Ahmed
SE-19028

LAB # 8

Interaction Diagrams

Object:
To understand Sequence Diagram Conceptual View



Introduction:
Sequence diagram and collaboration diagrams- both of which are called interaction diagrams.These two diagrams are used in the UML for modeling the **dynamic aspects of systems.** An interaction diagram shows an interaction, consisting of a set of objects and their relationships including messages that may be dispatched among them. An interaction diagram that emphasizes the time ordering of messages called Sequence **diagram.** An interaction diagram that emphasizes the structural organization of the objects that send and receive messages called **collaboration diagram.**

Contents:
Interaction diagram commonly contain

- Objects
- Links
- Messages

Interaction diagram may also contain notes and constraints.

1. Sequence Diagram:

Sequence Diagrams describe interactions among classes. These interactions are modeled as exchanges of messages.

- These diagrams focus on classes and the messages they exchange to accomplish some desired behavior.
- Sequence diagrams are a type of interaction diagrams. Sequence diagrams contain the following elements:

Class roles: which represent roles that objects may play within the interaction.
Lifelines: which represent the existence of an object over a period of time.
Activations: which represent the time during which an object is performing an operation.

Uses: To model the flows of control by time ordering.

Construction of a Sequence diagram:

Identify the workflows that you model as separate sequence diagrams.
Expose the objects for the individual sequence diagrams.

Include messages and conditions in the order of control flow for your individual Sequence diagrams.

Modeling Techniques: To model a flow of control by time ordering

Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.

Set the stage for the interaction by identifying which objects play a role in the Interaction. Expose the objects on the sequence diagram from left to right, placing the more important objects to the left and their neighboring objects to the right.

Set the lifeline for each object.

Starting with the message that initiates this interaction, Expose each subsequent message from top to bottom between the lifelines. Showing each message‘s properties, as necessary to explain the semantics of the interaction.

If you need to visualize the nesting of messages or the points in time when actual computation is taking place, represent each object‘s lifeline with its focus of control.

If you need to specify time or space constraints, represent each message with timing mark and Attach suitable time or space constraints.

If you need to specify this flow of control more formally, attach pre and post conditions to each message.

Kabeer Ahmed
SE-19028

LAB # 9

Interaction Diagrams

Object:
To understand Sequence Diagram illustrative View

Illustrative View

- In illustration view we explain the designing or construction of a sequence diagram.
- In illustration view we discuss the Components which are used in construction of Sequence Diagram.
- Notations and symbols which are used in Sequence Diagram.

A Sequence diagram has two dimensions

- Horizontal axis: represents different objects
- Vertical axis: represents time i.e shows the sequence of messages.

Components:

To understand what a sequence diagram is, you should be familiar with its components. Sequence diagrams are made up of the following elements:

Object Symbol

This box shape represents a class, or object, in UML. They demonstrate how an object will behave in the context of the system. Class attributes should not be listed in this shape.

Activation Box

Symbolized by a rectangle shape, an activation box represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.

Actor Symbol

Represented by a stick figure, actors are entities that are both interactive with and external to the system.

Package Symbol

Also known as a frame, this is a rectangle shape that is used in UML 2.0 notation to contain interactive elements of the diagram. The shape has a small inner rectangle for labeling the diagram.

Lifeline Symbol

A dashed vertical line that represents the passage of time as it extends downward. Along with time, they represent the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.

Option Loop Symbol

A rectangle shape with a smaller label within it. This symbol is used to model "if then" scenarios, i.e., a circumstance that will only occur under certain conditions.

Alternative Symbol

Used to symbolize a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

Sequence Symbols:

Packets of information that are transmitted between objects. They may reflect the start and execution of an operation, or the sending and reception of a signal.

Synchronous Message Symbol

Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.

Asynchronus Message Symbol

Represented by a solid line with a lined arrowhead. Asynchronous messages are those that don't require a response before the sender continues. Only the call should be included in the diagram.

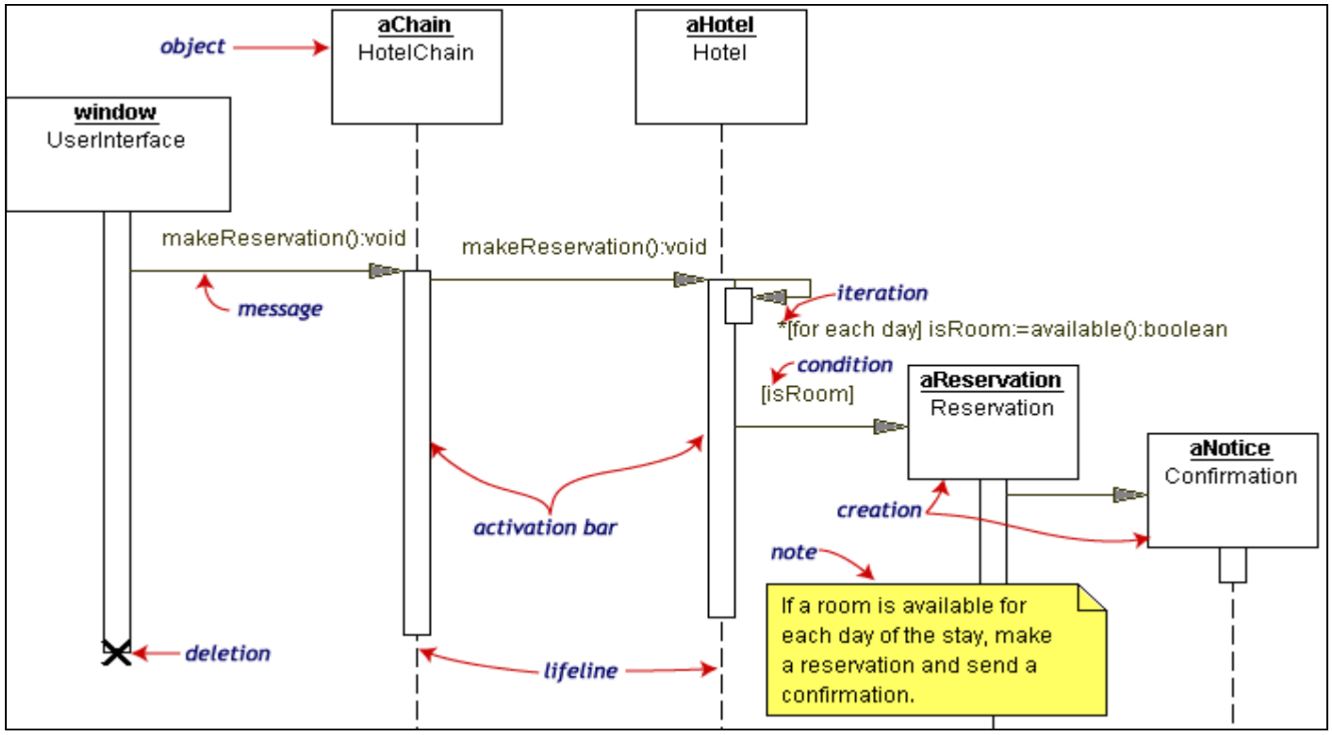
Kabeer Ahmed
SE-19028

LAB#09

Exercise:

Draw a sequence diagram for Hotel Reservation System

Hint: The object initiating the sequence of messages is a Reservation window. The Reservation window sends a makeReservation() message to a HotelChain. The HotelChain then sends a makeReservation() message to a Hotel. If the Hotel has available rooms, then it makes a Reservation and a Confirmation.

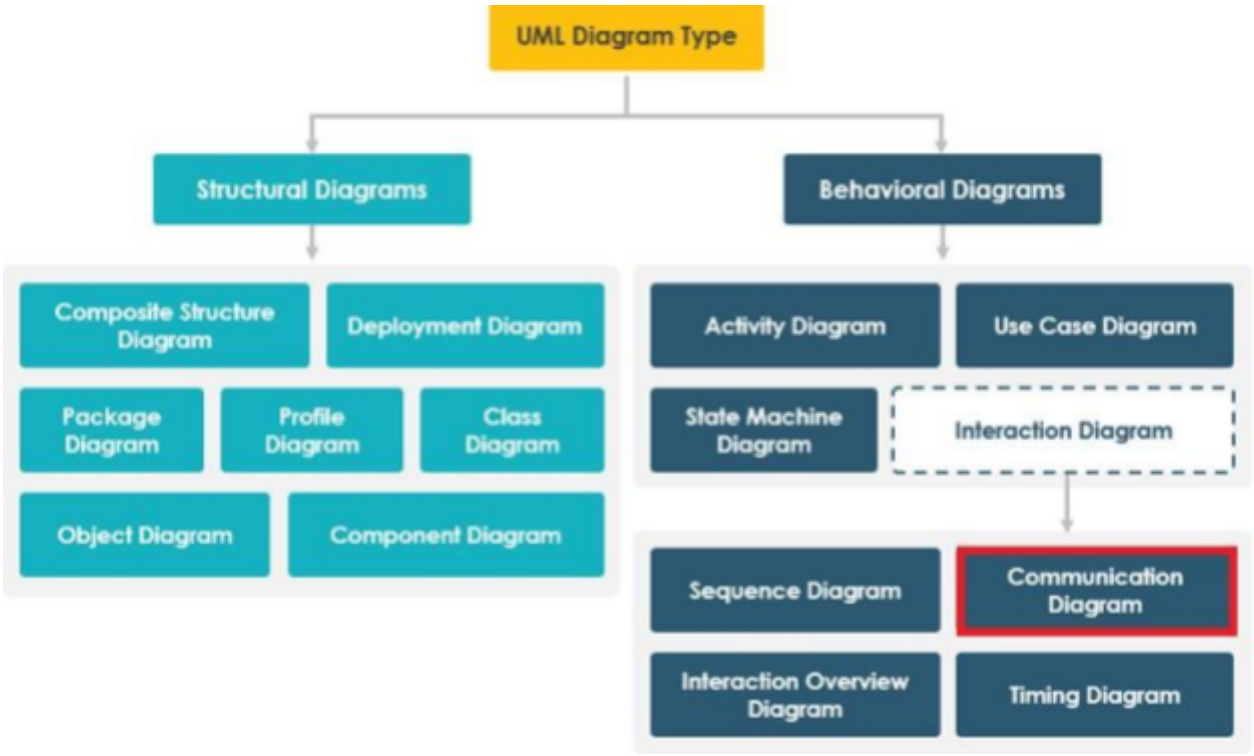


Kabeer Ahmed
SE-19028

LAB # 10

Interaction Diagrams

Object:
To understand Collaboration Diagram View



Collaboration Diagram:
A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages called **collaboration diagram.**

Construction

- First identify the objects that participate in interaction. And adorn them as the vertices in a graph
- Secondly establish the connection between the objects using links .
- Finally, adorn these links with the messages that objects send and receive messages.

Collaboration diagram has two features that distinguish it from sequence diagram.

1. **Path:** How one object is linked to another object. We can attach a path stereo type to the far end of a link.
2. **Sequence Number:** It indicates the time order of messages.

Uses:
To model the flows of control by organization.

Modeling Techniques:

To model flow of control by organization:

Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.

Set the stage for the interaction by identifying which objects play a role in the interaction. Expose them on the collaboration diagram as vertices in a graph, placing the more important objects in the center of the diagram and their neighboring objects to the outside
Set the initial properties of each of these objects.

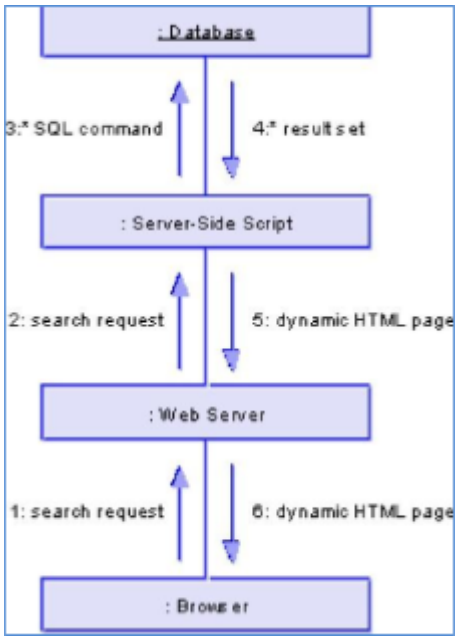
Specify the links among these objects, along which messages may pass.

1. Expose association links first; these are most important ones, because they represent structural connections.
2. Expose the other links if necessary and adorn them with suitable stereotypes.

Starting with the message that initiates this interaction, attach each subsequent message to the appropriate link. Setting its sequence number, as appropriate. Show the nesting sequence using decimal numbering.

If you need to specify time or space constraints, represent each message with a timing mark and Attach suitable time or space constraints.

If you need to specify this flow of control more formally, attach pre and post conditions to each message.



Kabeer Ahmed
SE-19028

LAB#10

Exercise:

1. Draw a collaboration diagram for Hotel Reservation System

Hint: The object initiating the sequence of messages is **a Reservation window.** The **Reservation window** sends a makeReservation() message to a **HotelChain.** The **HotelChain** then sends a makeReservation() message to a **Hotel.** If the **Hotel** has available rooms, then it makes a **Reservation** and a **Confirmation.**