Software Engineering

Unit 5: Object Oriented Design and Implementation

Object Oriented Design and Implementation

5.1. Need of design phase

5.2. Interaction diagrams

5.3. Sequence diagrams

5.4. Collaboration diagrams

5.5. Activity diagrams

5.6. State chart diagrams

5.7. Object-oriented design principles for improving software quality

CE: 5.5
Activity diagrams

CE: 5.5 Activity diagrams

- The *dynamic phases* of a system are modelled through ***activity diagrams***.
- Activity diagrams are used to model a system to show *the working of*...
  a process or ***a use case*** or ***an operation***.
- That means, the activities carried out in ***a process/workflow*** or ***an operation*** are represented in an activity diagram.
- Activity diagrams enable to visualize, understand and document the flow of activities in…
  a process or ***a use case*** or ***an operation***.
- They represents executable computational steps in the system.
- They explore the order in which the activities must be carried out to achieve a goal.

CE: 5.5 Activity diagrams (Conti…)

- Activity diagrams have many applications by using the following:

  1. To increase the understanding of a business model or a process or a use case.
  2. To determine the objective for business-relation e-commerce applications.
  3. To simplify a given operation.

- In short, ***activity diagrams*** show the steps which are involved in ***a process*** or ***a use case*** or ***an operation***.

CE: 5.5 Activity diagrams (Conti…)

- The various UML notations used for activity diagrams are:

Start

The *start* state represents beginning of an activity diagram.

Stop

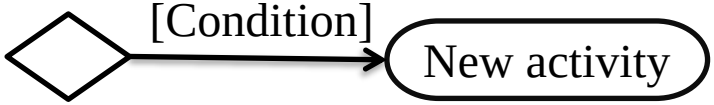
The *stop* state represents end of an activity diagram.

Activity

An *activity* is an executing set of steps that take place in an activity diagram.

Transition

***Transition* represents a path from one activity to another activity.**

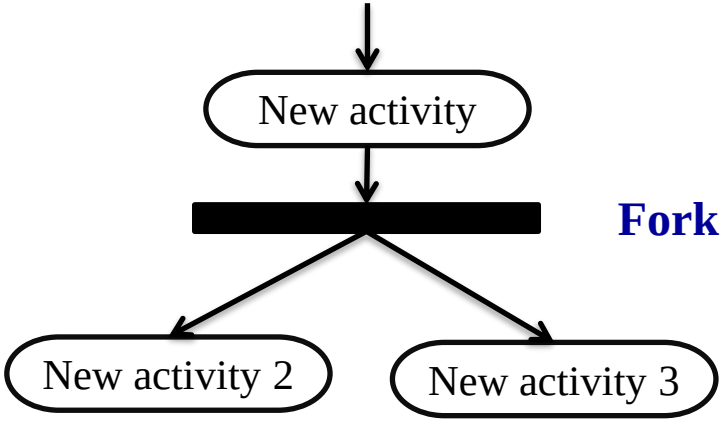


Branching

***Branching* is used to represent an if-else condition with a guard condition is square brackets.**

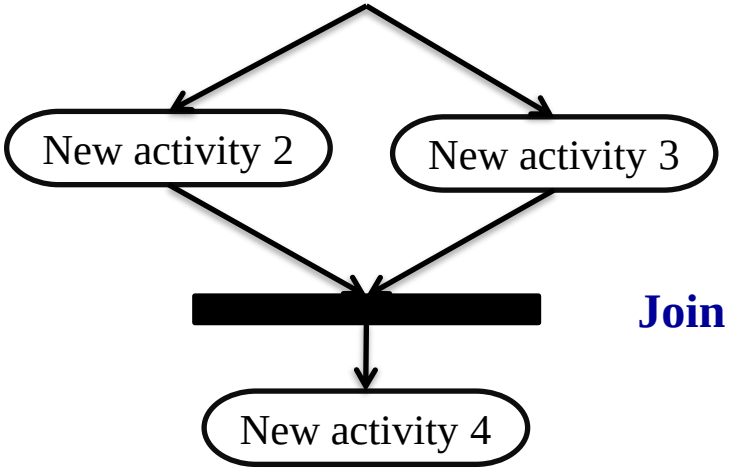
CE: 5.5 Activity diagrams (Conti…)

- The various UML notations used for activity diagrams are: (Conti…)



Fork

Fork **breaks up an activity into a set of concurrent sub-activities.**



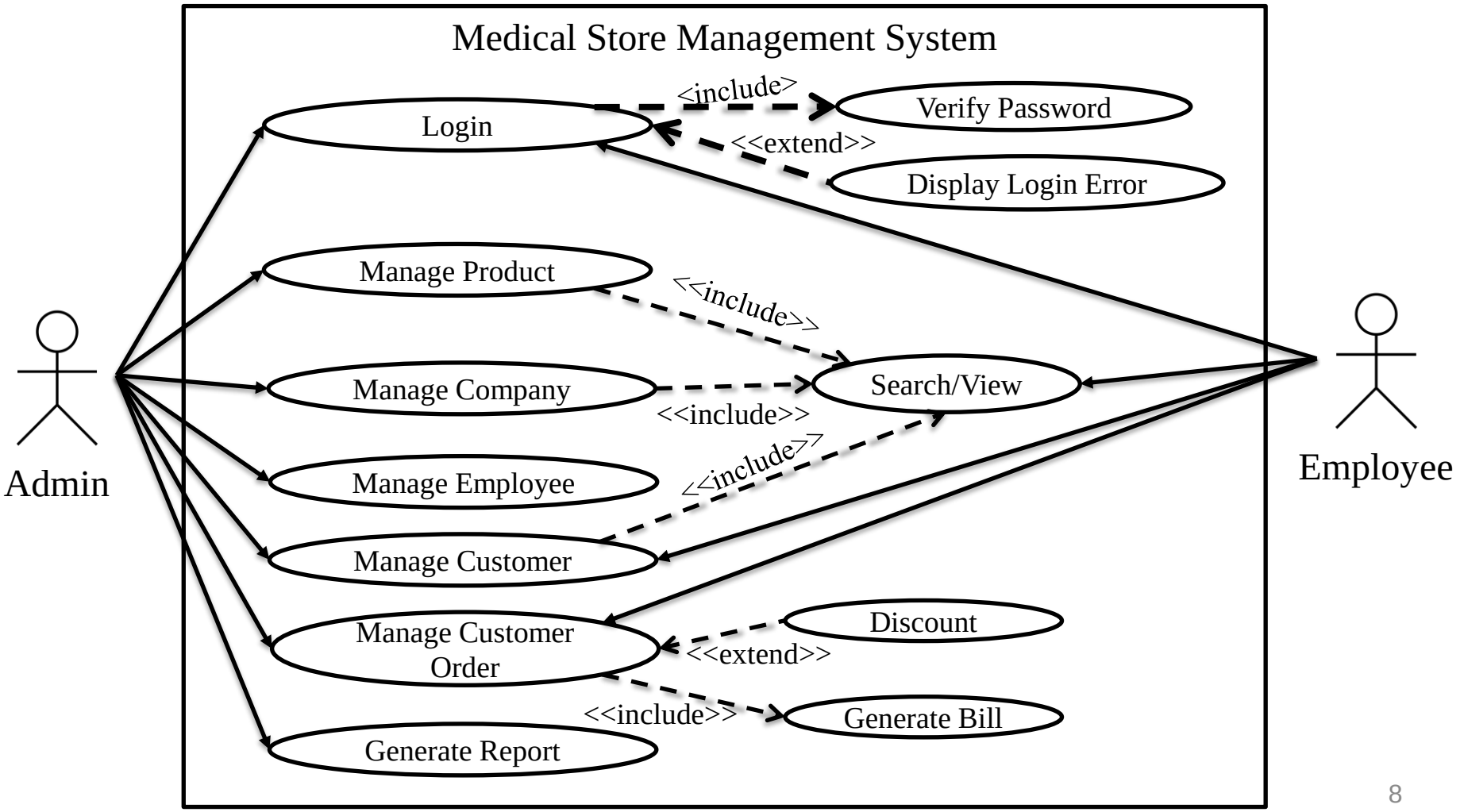
Join

Join **is used to combine the flow of concurrent activities into the next single activity.**

CE: 5.5 Activity diagrams (Conti…)

- For Example: Consider the Medical Store Management System.

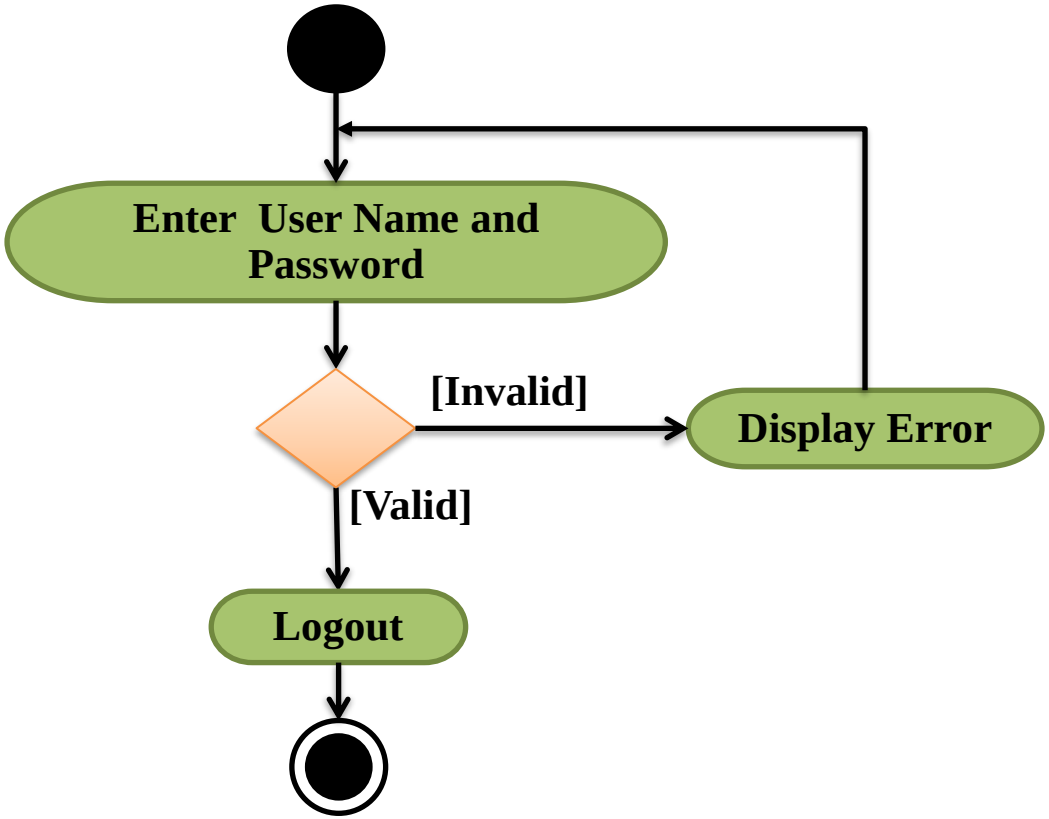
Use Case Diagram for Medical Store Management System



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a use case.

- For Example: **Activity diagram for "User Login" use case.**

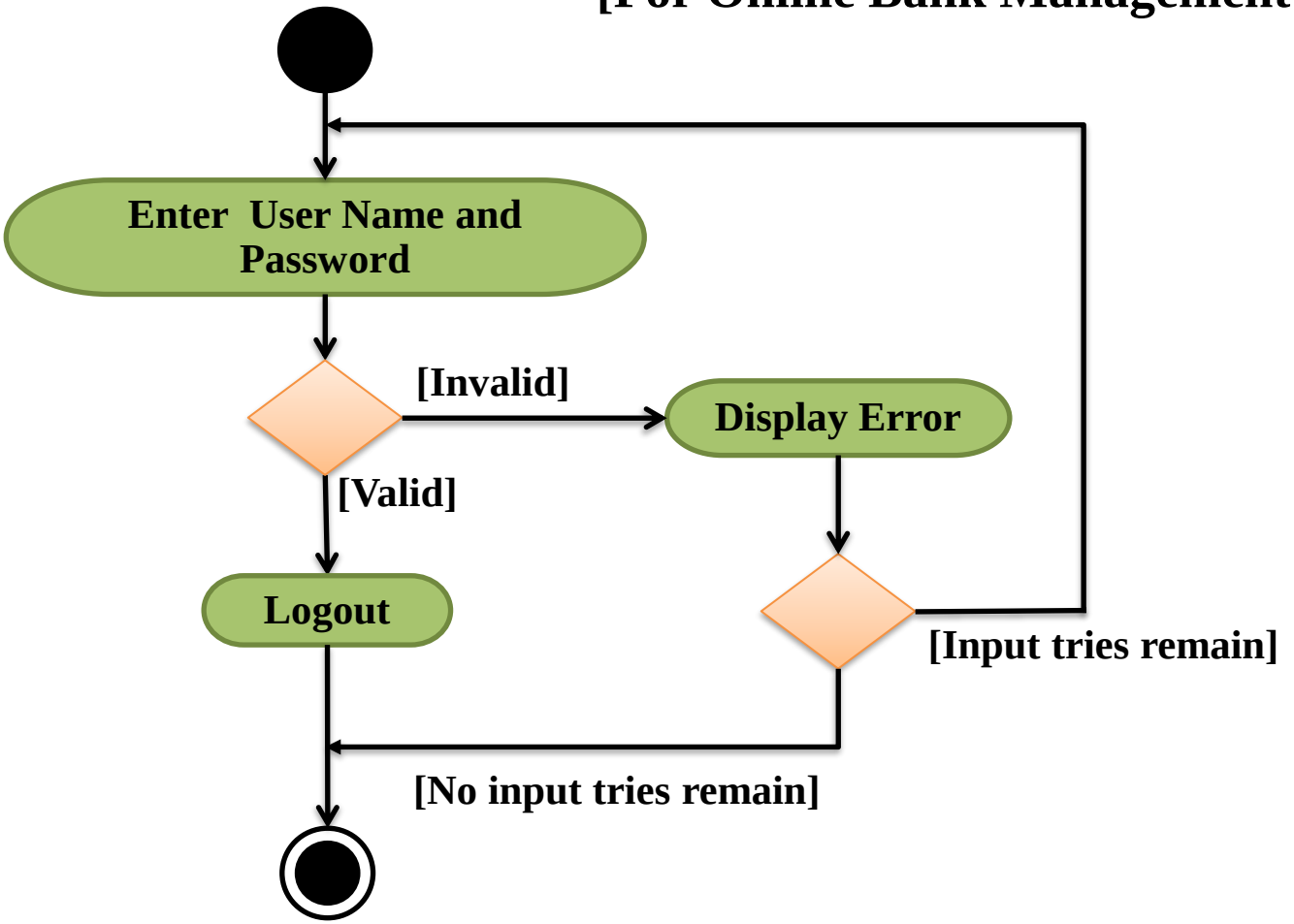


CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a use case.

- For Example: **Activity diagram for "User Login" use case.**

[For Online Bank Management System]

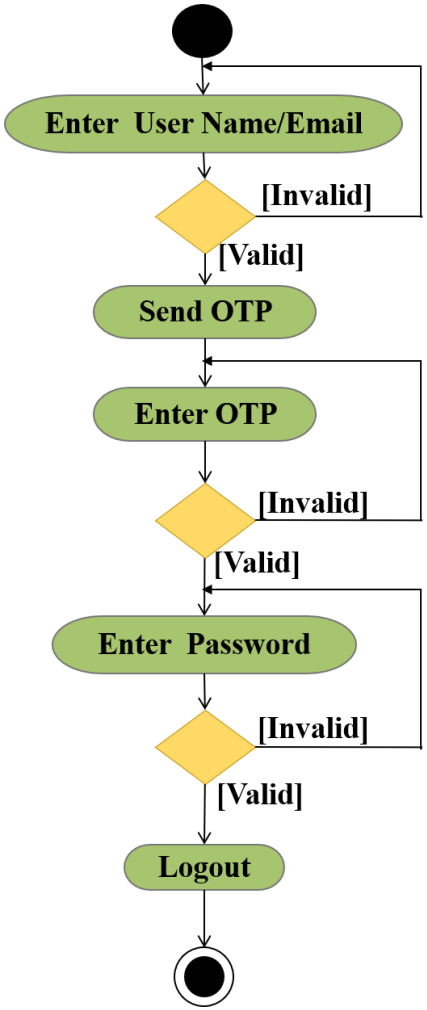


CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a use case.

- For Example: **Activity diagram for "User Login" use case.**

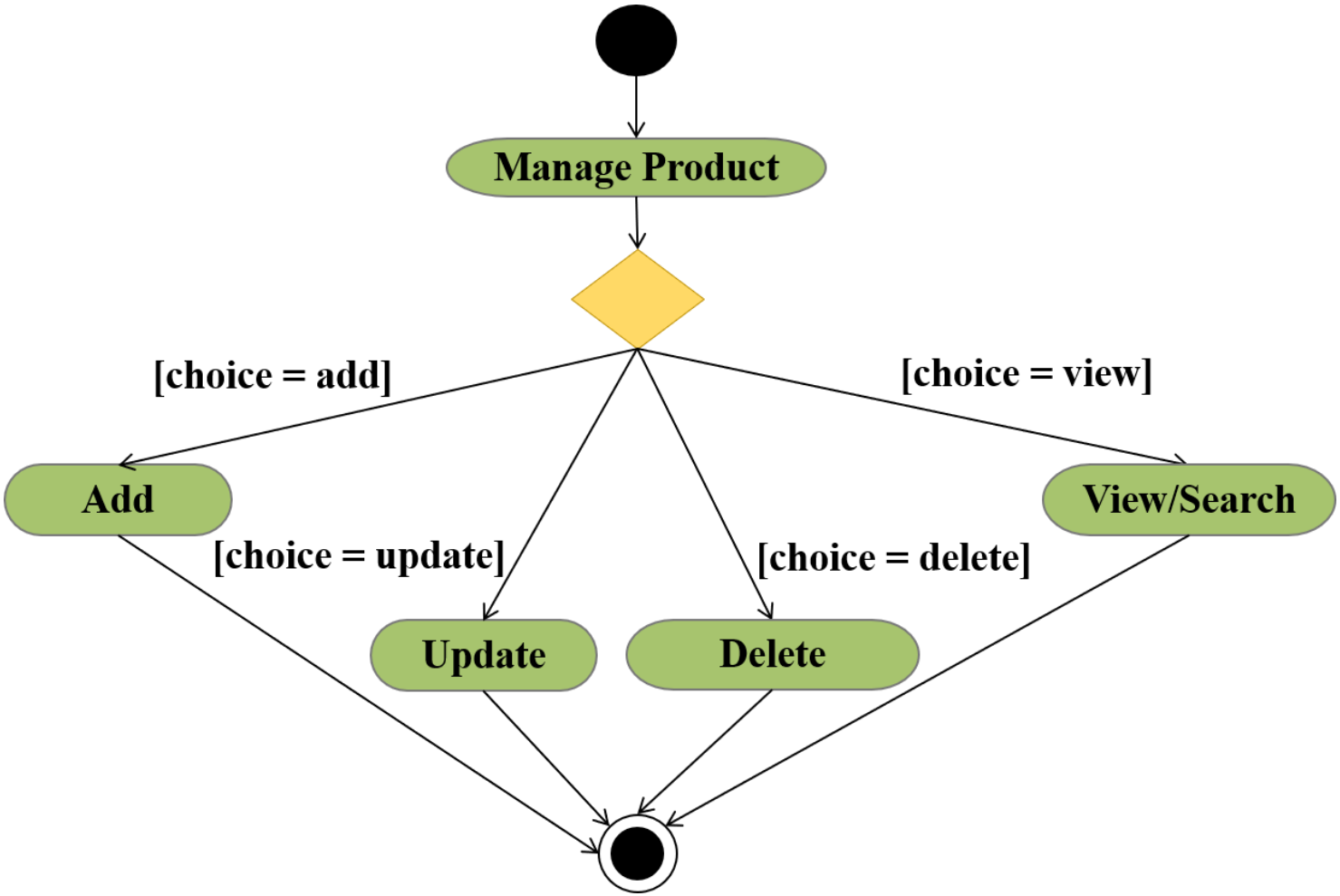
[For online site may be…]



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a use case.

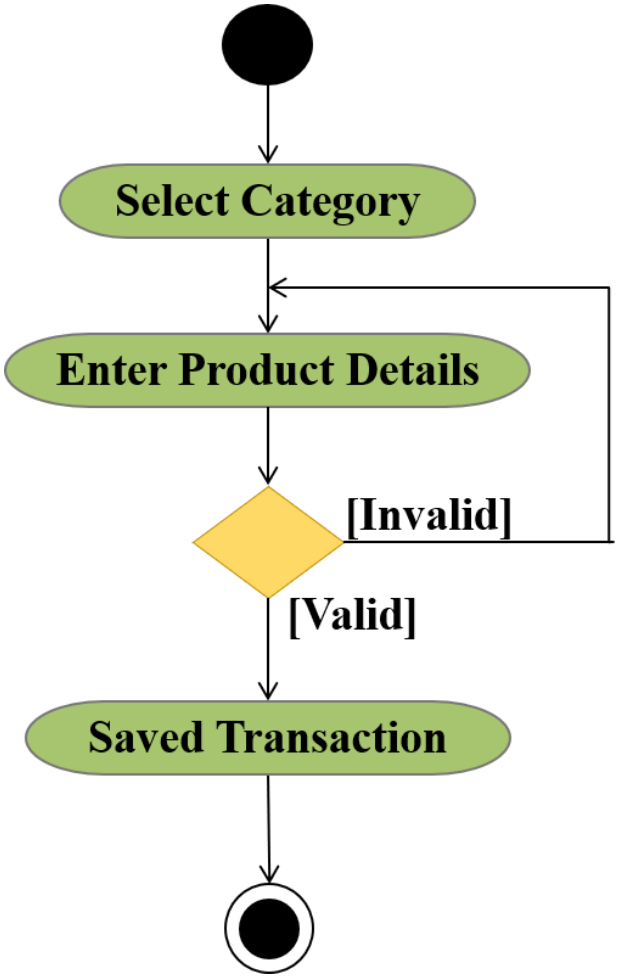
- For Example: **Activity diagram for "Manage Product" use case.**



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in an operation.

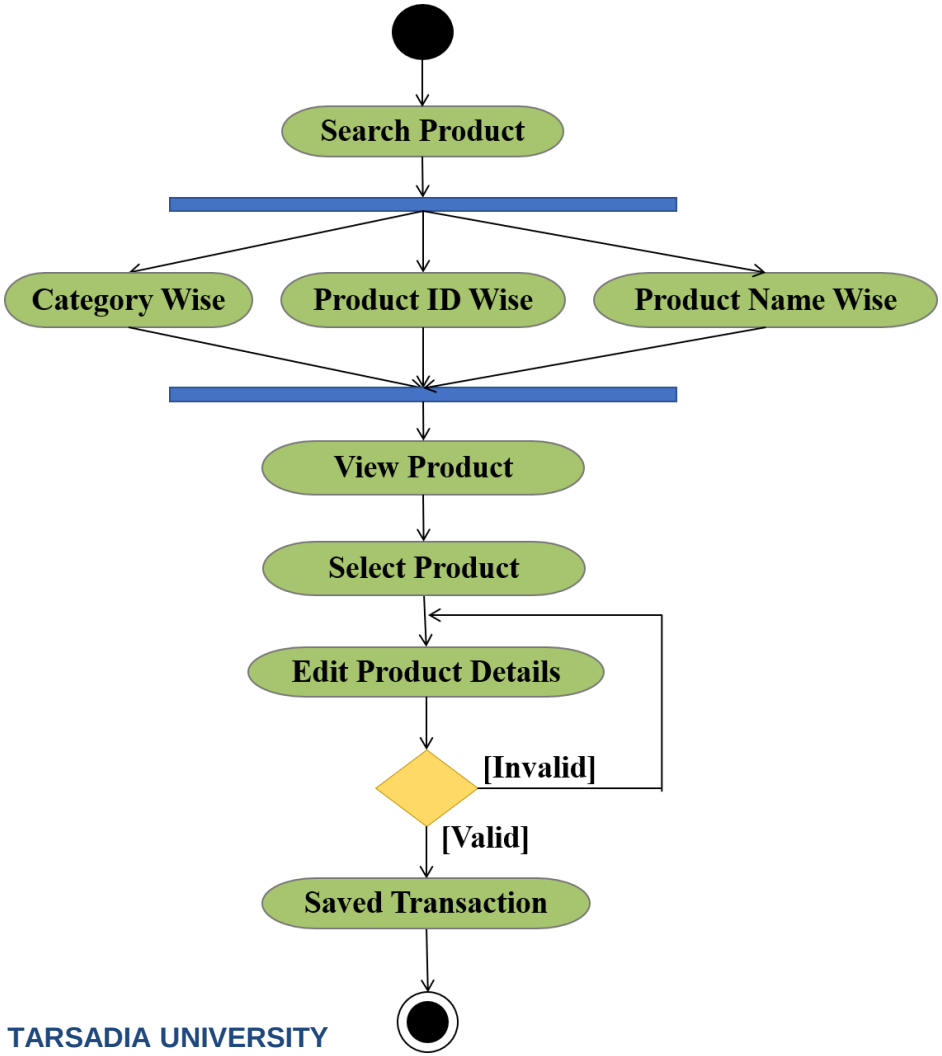
- For Example: **Activity diagram for "Add Product" operation.**



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in an operation.

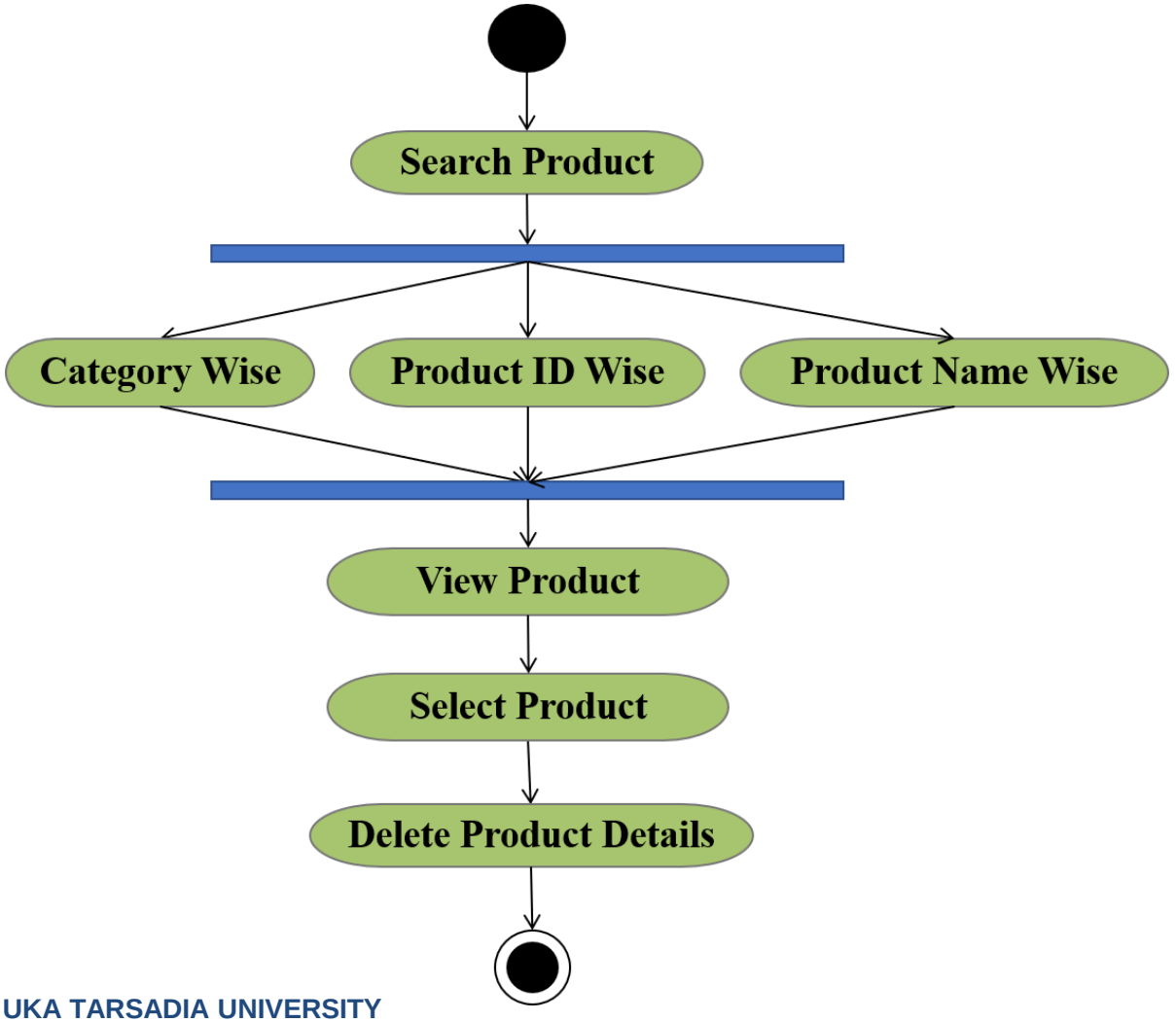
- For Example: **Activity diagram for "Update Product" operation.**



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in an operation.

- For Example: **Activity diagram for "Delete Product" operation.**

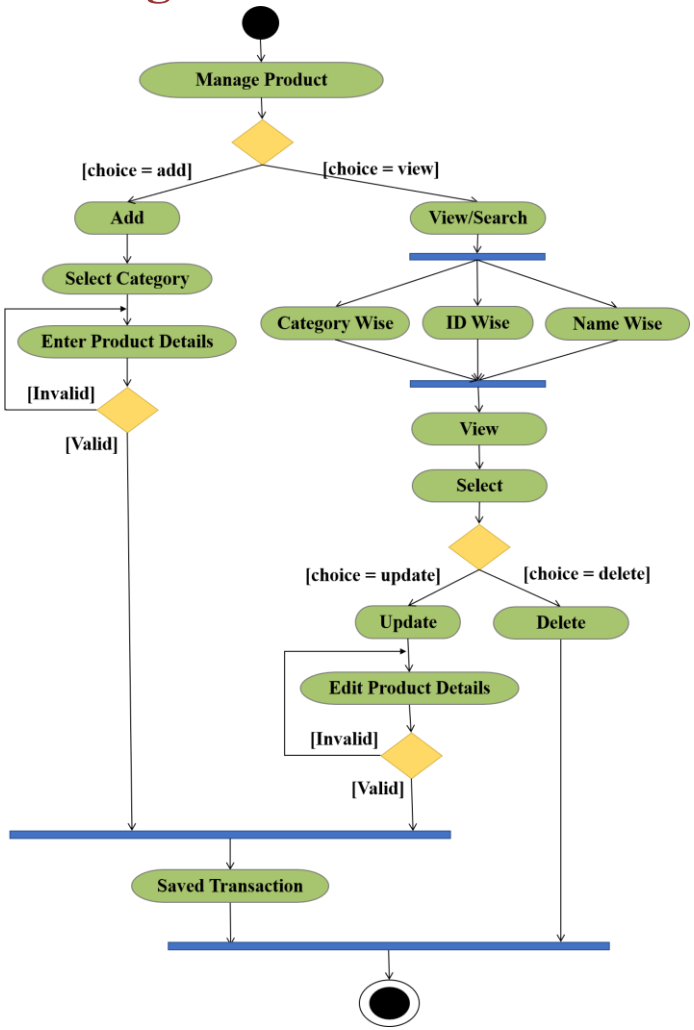


CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a use case.

- For Example: **Activity diagram for "Manage Product" use case.**

Activity diagram for Manage Product Link

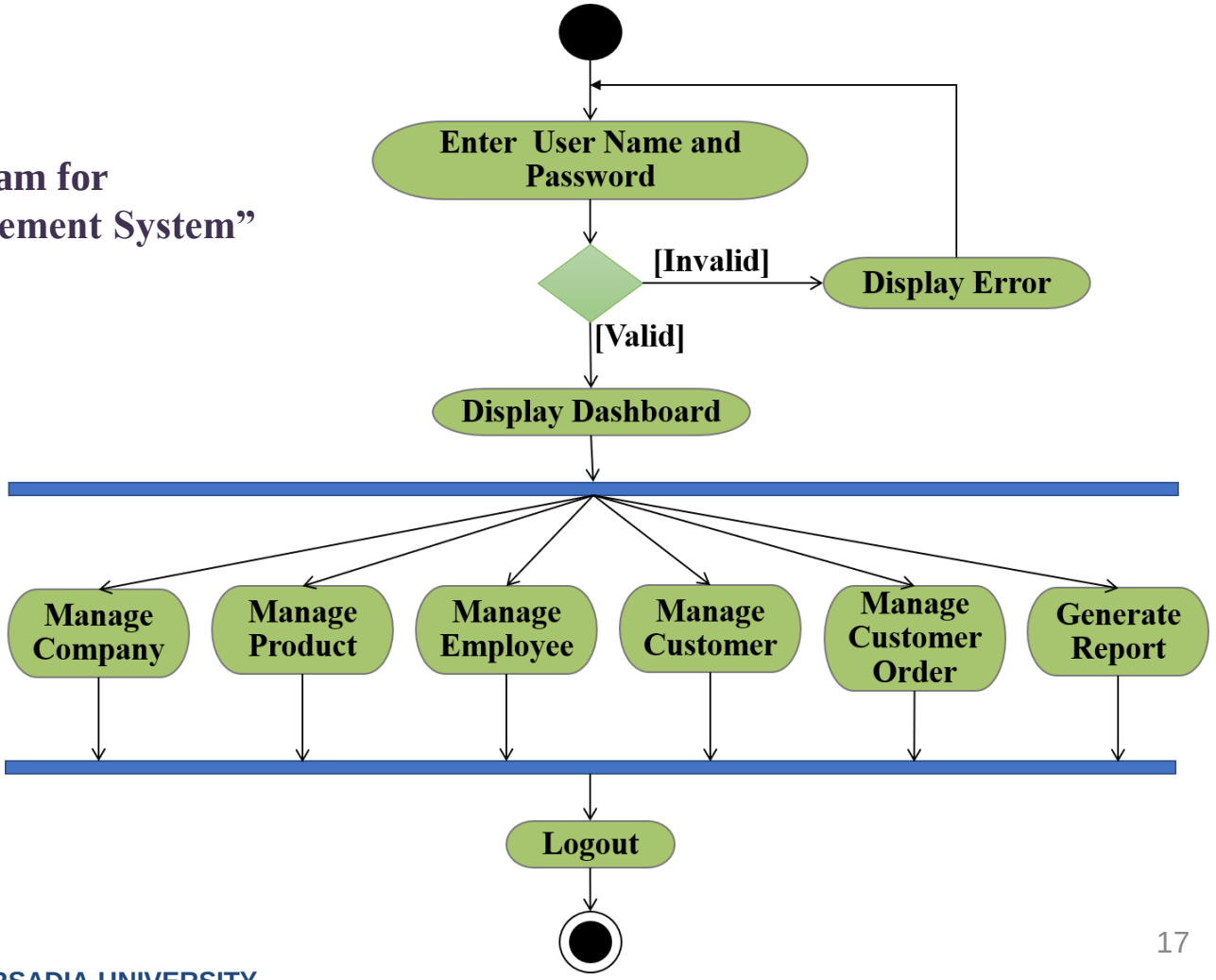


CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a process.

- For Example: **Activity diagram for "Medical Store Management System" process.**

Activity diagram for "Medical Store Management System" Link



CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a process.

- For Example: **Activity diagram for "Online Bakery Management System".**

Activity diagram for
"Online Bakery Management System"
Link

CE: 5.5 Activity diagrams (Conti…)

- But, what will you do, if the system is used by multiple users?
- ***Activity diagrams*** tell us what happens, but they do not tell us **who** does **what**.
- In programming, this means that the *diagram does not convey which class is responsible for each activity*.
- One way around this is to label each activity with the responsible *class* or *human*. This works, but does not offer the same clarity as *interaction diagrams* for showing communication among objects.
- ***Swimlanes*** are a way around this.

CE: 5.5 Activity diagrams (Conti…)

What is Swimlanes?

- **A *swimlane* groups all the activities that are carried out by the particular class.**
- ***Swimlanes*** are represented by ***vertical lines that are divided in each group***.
- ***Each group*** represents ***the responsibilities of a particular class.***
- ***Each swimlane*** is represented by a ***unique name***.
- The ***transitions*** may cross between classes or entities in the swimlanes.

CE: 5.5 Activity diagrams (Conti…)

- For Example:

Consider an E-commerce application for purchasing an item from the Internet. Prepare an activity diagram by taking the help of swimlanes and following steps:

1. The customer selects the items to be purchased.
2. The customer adds the items in the shopping cart.
3. The customer places an order.
4. The vender receives the order.
5. A secure payment gateway asks for the customer's debit/credit card details.
6. The customer provides his/her details to the payment gateway.
7. The credit card company verifies the customer's details and approves the transaction.
8. The vender confirms the order.
9. The vendor communicates with warehouse that ships the order to the customer at the specified shipping address.
10. The credit card company sends the bill to the customer for payment.

CE: 5.5 Activity diagrams (Conti…)

Activity diagram to show the steps which are involved in a process.

- For Example: **Activity diagram for E-commerce application.**

Activity diagram for
E-commerce application
Link

CE: 5.5 Activity diagrams (Conti…)

- Activity diagrams are the most important UML diagrams for doing ***business process modeling***.
- In software development, it is generally used to describe the flow of different activities and actions.
- These can be both sequential and in parallel. They describe the objects used, consumed or produced by an activity and the relationship between the different activities.
- All the above are essential in ***business process modeling***.

CE: 5.6
Statechart Diagrams

CE: 5.6 Statechart Diagrams

- Like activity diagrams, ***statechart diagrams*** are also modelled for the *dynamic phases* of the system.
- The name of the diagram itself clarifies, the purpose of the diagram and the other details.
- It describes different ***states*** of ***an object*** or a component in a system.
- The *states* are specific to an object of a system.
- The difference between Statechart Diagrams and Activity Diagrams is…

Statechart Diagrams	Activity Diagrams
1. They are based on states. 2. It models the states of an object's lifetime.	1. They are based on activities. 2. It is used to model the sequence of activities in a process or an operation.

CE: 5.6 Statechart Diagrams (Conti…)

- They help the developers to obtain a clear understanding of the dynamic behaviour of an object.
- And, they are very useful for modelling the reactive (responsive) system.

CE: 5.6 Statechart Diagrams (Conti…)

- The various UML notations used for statechart diagrams are:

Start state

The *start state* represents beginning of a statechart diagram, which specifies the creation of the object.

Stop state

The *stop state* represents end of a statechart diagram, which specifies the destruction of the object.

State

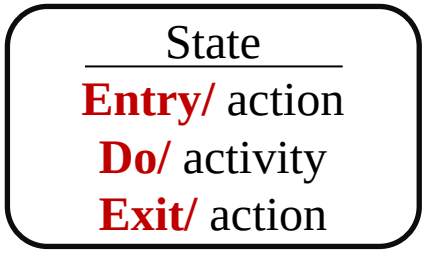
Rectangle with rounded ends.

- ✓ **An *state* represents one of the conditions or the situations of an object, that may satisfy during its lifetime.**
- ✓ **The *state* of the object may be determined by…**
  - *the object that may perform some activity or*
  - *the object that may wait for some event to happen*.

CE: 5.6 Statechart Diagrams (Conti…)

- The various UML notations used for statechart diagrams are: (Conti…)

State Conti…



State with entry and exit actions

✓ **The *start* may also consist of…**

1. **Entry action,**
2. **Do activity and**
3. **Exit action.**

✓ The ***entry action*** takes place, when the object enters a given state.

✓ The ***do activity*** specifies, the tasks/activities that must be performed; while in the current state and continues until the ***state is exited***.

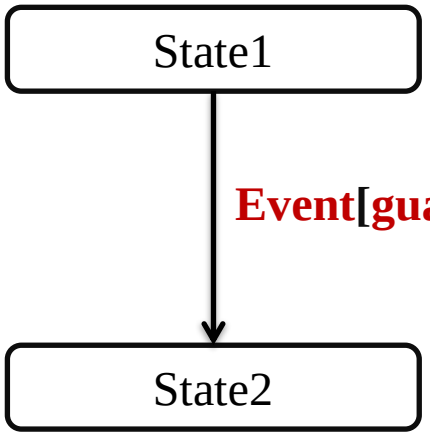
✓ The *entry* and *exit* actions are preceded by a word *entry/exit followed by a* '/' (slash).

CE: 5.6 Statechart Diagrams (Conti…)

- The various UML notations used for statechart diagrams are: (Conti…)

State Transition

- ✓ ***A state transition* represents *the change of an object* from *one state* to *another state*.**
- ✓ **It may be associated with…**
  1. **an event or**
  2. **an action or**
  3. **a guard condition.**



State Transition in detail

CE: 5.6 Statechart Diagrams (Conti…)

- The various UML notations used for statechart diagrams are: (conti…)

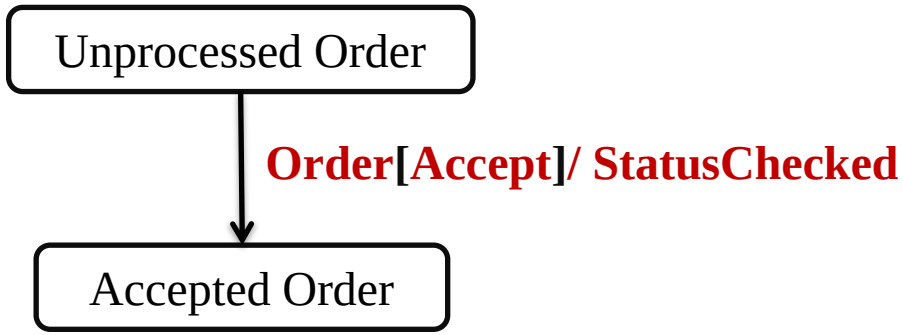
1. Event:

✓ An ***event*** is a message that is from one object to another object.

✓ In the statechart diagram, an ***event*** can be shown using the operation name on the *state transition*.

✓ The operation may consist of arguments or may be a message from some other object.

✓ For Example: For the process of order.



CE: 5.6 Statechart Diagrams (Conti…)

- The various UML notations used for statechart diagrams are: (conti…)

2. Action:

✓ An ***action*** is a behaviour that occurs in response to a *state transition*.

✓ The ***actions*** become the operation of the object.

✓ All the ***actions*** are non-interruptible, so it is separated by '/' (slash) after the ***event name***.

3. Guard Condition:

✓ A ***guard condition*** is a ***boolean expression*** that tests, whether the condition is true or not.

✓ If the condition is true, then the *state transition* is done, otherwise not.

CE: 5.7
Object-oriented design principles for improving software quality

CE: 5.7 Object-oriented design principles for improving software quality

- Coad and Yourdon (1991) defined a set of design principles for improving the quality of a software product.

- These principles will result in the development of a software product with better object-oriented design and also to produce a maintainable system.

- The provided principles/guidelines include…
  1. Cohesion
  2. Coupling
  3. Design clarity
  4. Class hierarchy depth
  5. Simple classes and objects

CE: 5.7 Object-oriented design principles for improving software quality (Conti…)

1. Cohesion:

- The attributes and operations in a class should be highly cohesive.
- Each operation should be designed to fulfil one single purpose.

2. Coupling:

- Interaction coupling between classes should be kept minimum.
- The number of message (sent and received) between classes should be reduced.
- Inheritance-based coupling between classes should be increased.

3. Design clarity:

- The vocabulary used should be correct, concise, unambiguous and consistent.

CE: 5.7 Object-oriented design principles for improving software quality (Conti…)

3. Design clarity: (Conti…)

- The names of the classes along with their attributes and operations should convey their intended meaning.

- The meaning and purpose of the class along with its responsibilities should be clear.

4. Class hierarchy depth:

- The concept of generalization-specialization should be used, wherever it is necessary.

- In other words, unnecessary and excessive use of inheritance (only for the sake of increasing reusability) should be avoided.

- Inheritance hierarchy should represent solution to a problem.

CE: 5.7 Object-oriented design principles for improving software quality (Conti…)

5. Simple classes and objects:

- Excessive attributes should be avoided in a class.

- Class definition should be simple, clear, concise and understandable.


Thank You