SS ZG514
Object Oriented Analysis and Design

Dr. Ritu Arora
rituarora@pilani.bits-pilani.ac.in

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Content s of these slides are adapted from Applying UML and Patterns, Craig Larman, 3rd edition

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

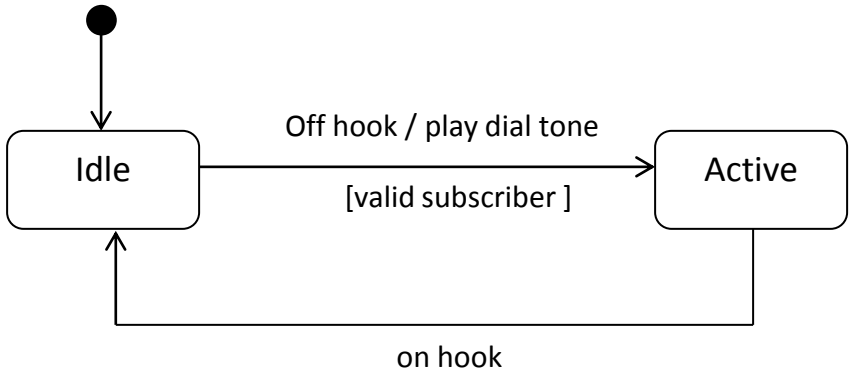
UML State Machine Diagrams

State Machine Diagram

- The state machine diagram (also known as state transition diagram) illustrates the events and states of things.
- It can be drawn for a System, subsystem, or an Object of the System.
- It usually depicts the lifecycle of an object; events that it experiences, its transitions, and states it is in between these events.

- An **event** is a trigger or an occurrence.
- A **state** is the condition of an object at a moment in time - the time between events.
- A **transition** is a relationship between two states that indicates that when an event occurs, the object moves from the prior state to the subsequent state.

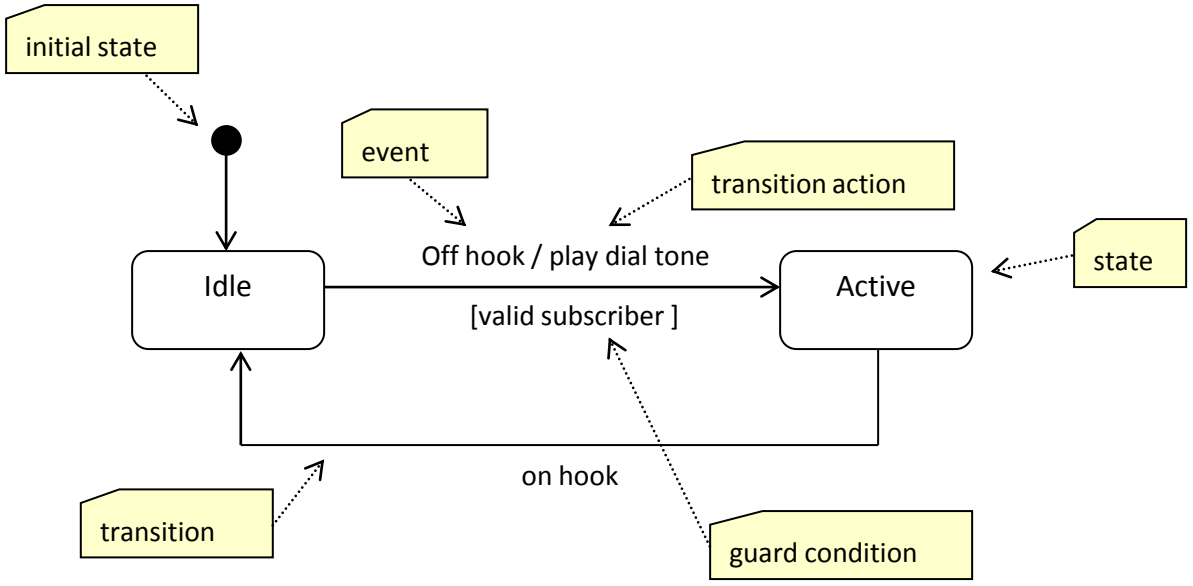
State Machine Diagram: Telephone System



State Machine Diagram for Telephone

State Machine Diagram: Notations

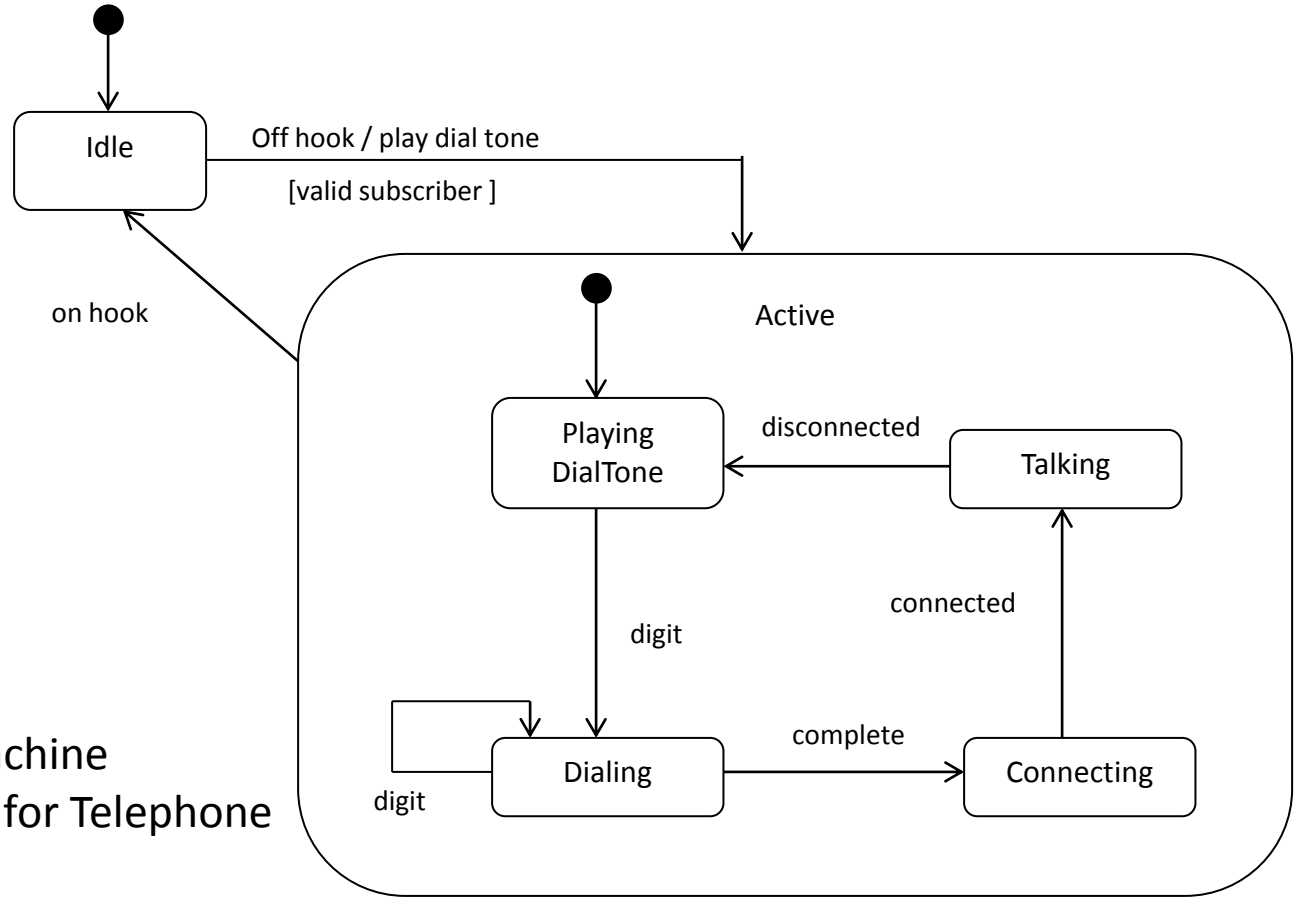
- States are shown as rounded-rectangles.
- Transitions are shown as arrows, labeled with their event.
- An initial pseudo-state is added, which automatically transitions to another state when the instance is created.



State Machine Diagram: Notations

State Machine Diagram: Notations

- Nested States: A state allows nesting to contain sub-states; a sub-state inherits the transitions of its super-state.



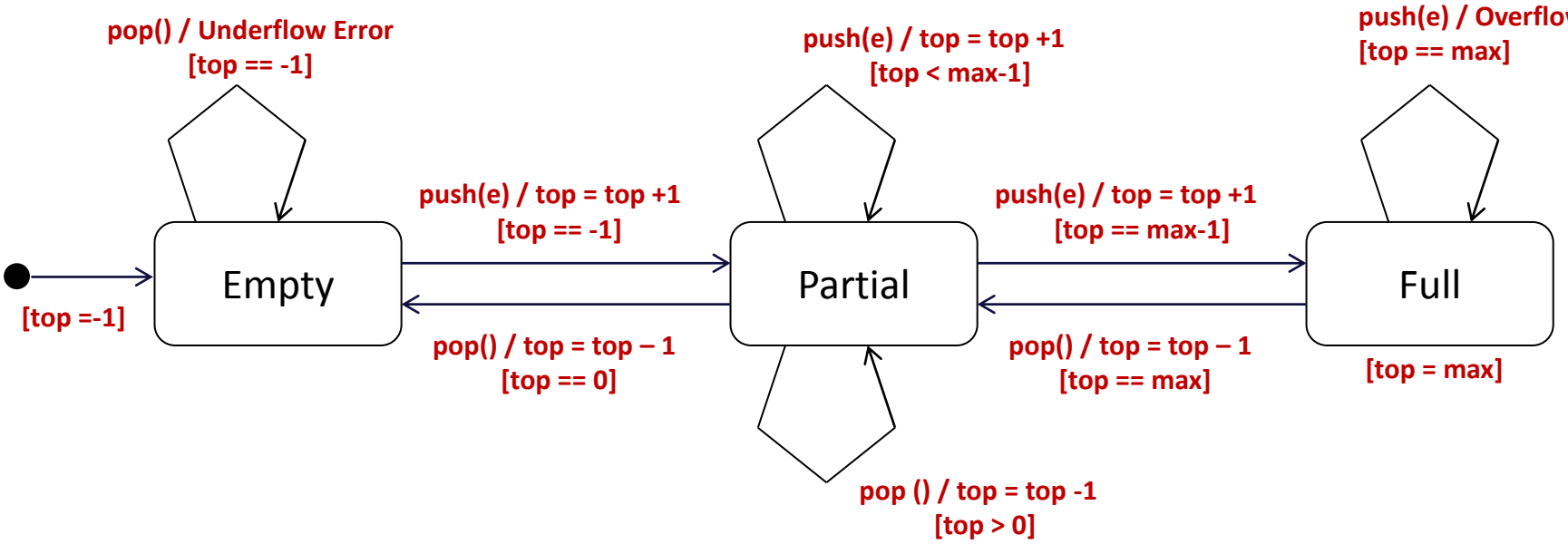
State Machine Diagram for Telephone

State Machine Diagram

Exercise:

- Draw the state machine diagram for a stack.

State Machine Diagram: Stack



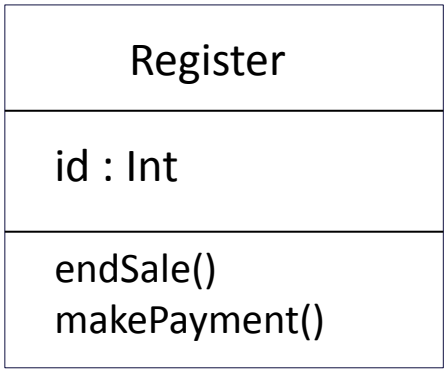
top =-1;
max =9;
Stack with 10 elements

UML Class Diagrams

Class Diagrams

- Used for static object modeling.
- Illustrates classes, interfaces and their associations.
- Design class diagram (DCD) – class diagram drawn in a software or design perspective.

- A rectangular box is used to represent a class or an interface. This rectangular box need to be partitioned into three sections: first section for class name, second for attributes and third for methods.



Class Diagrams: Representing Attributes

- Ways to represent attributes (also called, structural properties) of a class:
  - attribute text notation, e.g. *currentSale: Sale*
  - association line notation
  - Both

Format of the attribute text notation:

visibility name: type multiplicity = default {property-string}

Class Diagrams: Representing Visibility

UML provides four abbreviations for visibility:

+ (public)

- (private)

~ (package)

(protected)

- Attributes are assumed private if no visibility is shown

Class Diagrams: Representing Attributes

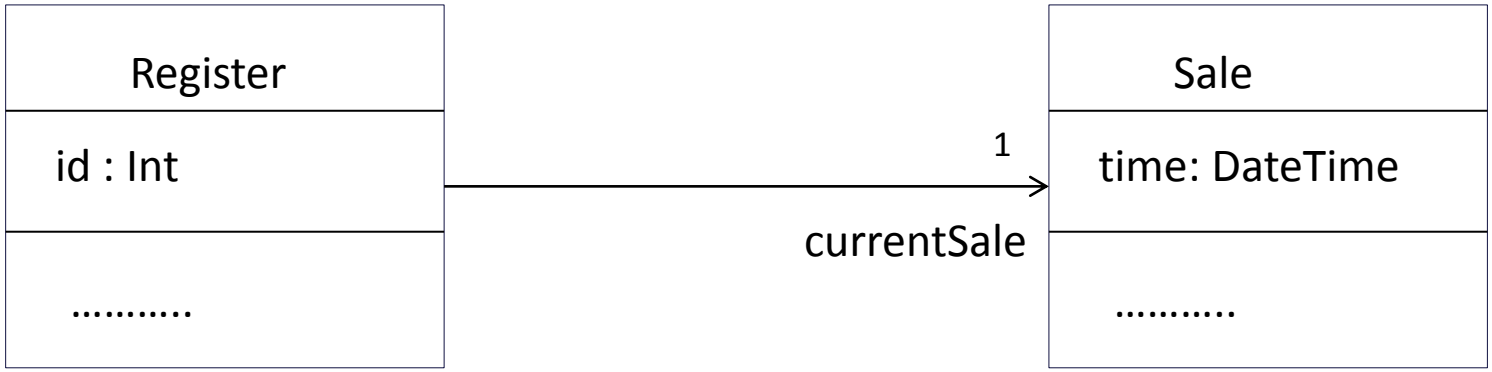
innovate achieve lead

Attribute text notation:

Class Diagrams: Representing Attributes

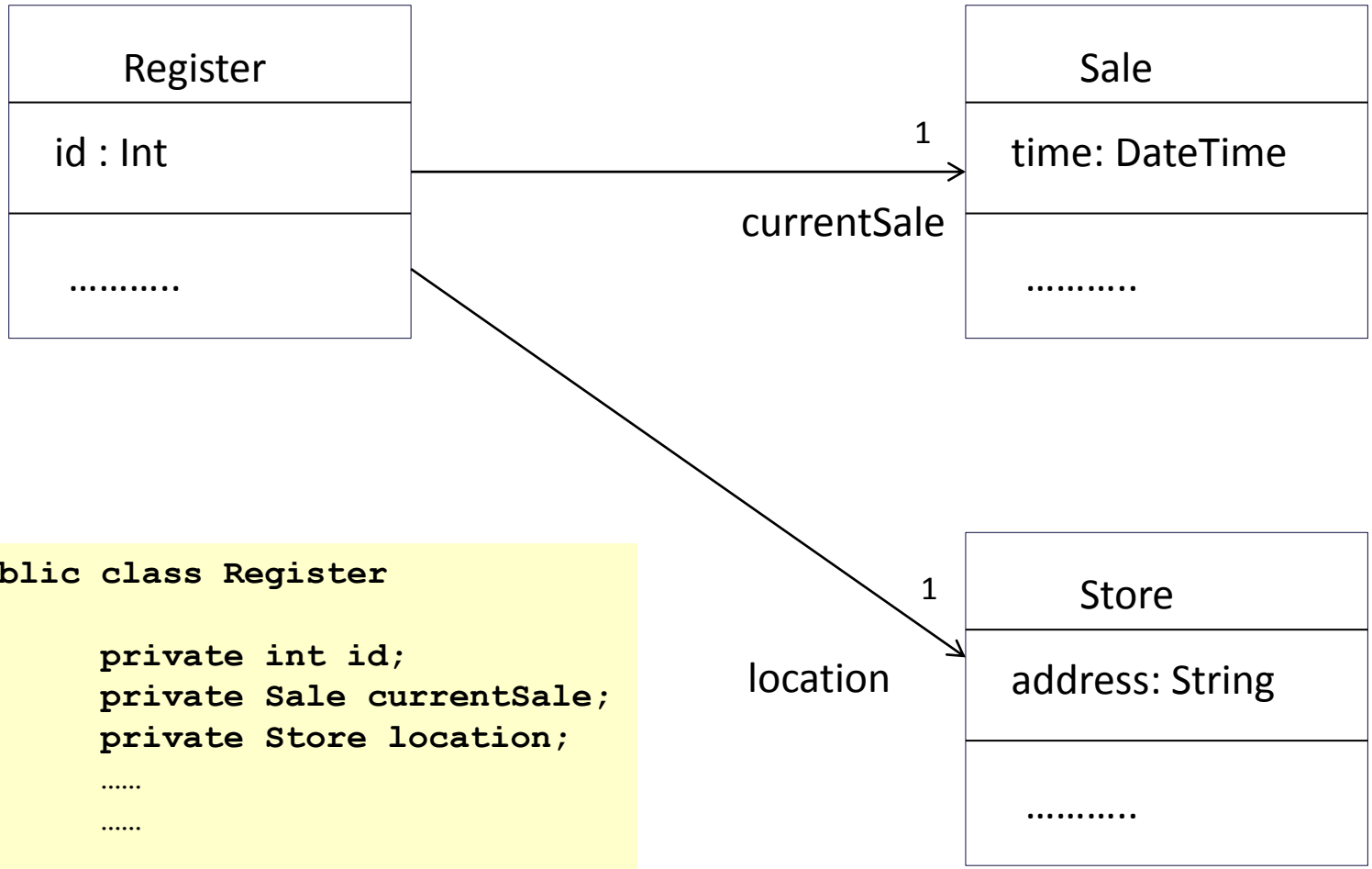
Attribute-as-association line has the following style:

- A navigability arrow pointing from the source to target.
- A multiplicity at the target end, but not the source end
- A rolename only at the target end to show the attribute name
- No association name



Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

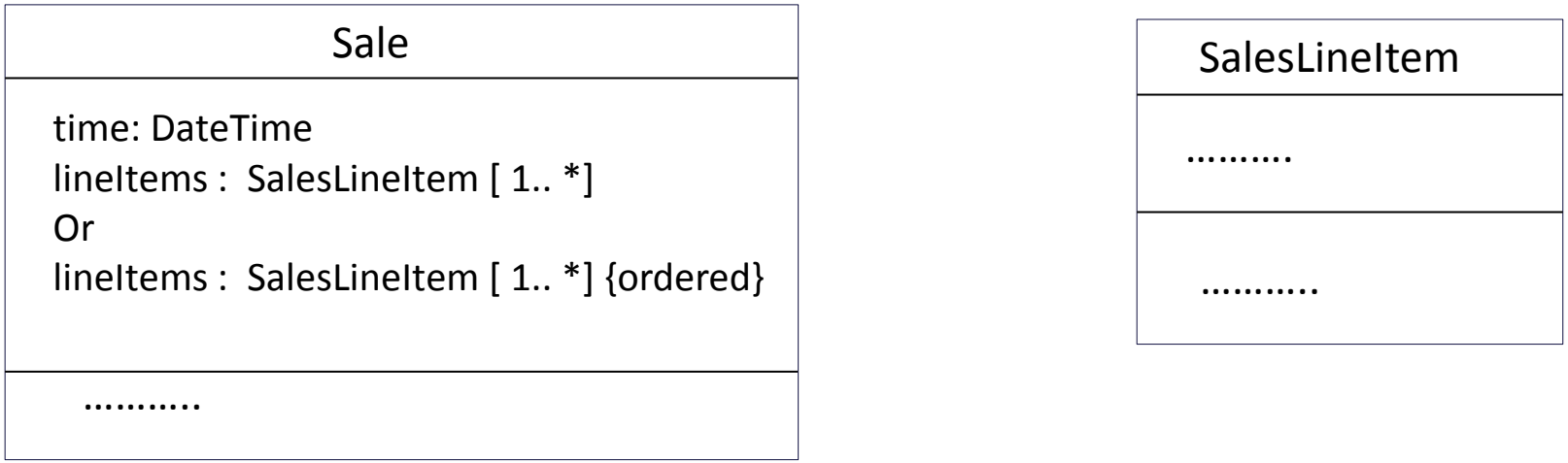
Class Diagrams: Representing Attributes

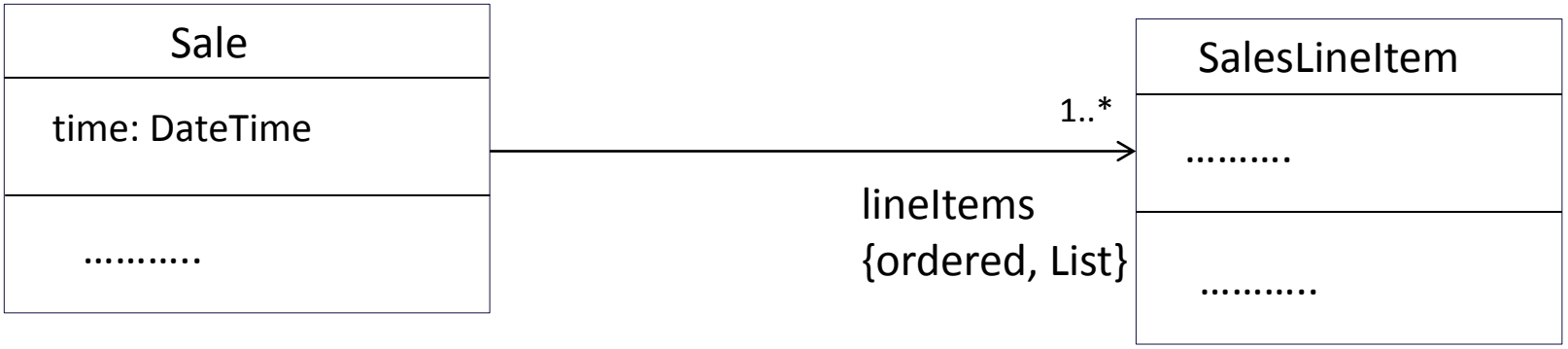



```
public class Register
{
        private int id;
        private Sale currentSale;
        private Store location;
        ……
        ……
}
```

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Collection Attributes





Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Operations

- Format for representing operations:
  visibility name (parameter-list) : return-type {property-string}

- Operations are assumed to be public, if no visibility is shown.
- The property-string contains arbitrary additional information, such as exceptions that may be raised, or if the operation is abstract.
- UML allows operation signature to be written in any programming language as well.
- A method is an implementation of an operation and represented using note.

Class Diagrams: Representing Associations

innovate achieve lead

- Association carry information about relationships among objects.
- Drawn using a solid line, without arrows.
- Associations have names (rolename) and multiplicity written at the association ends.

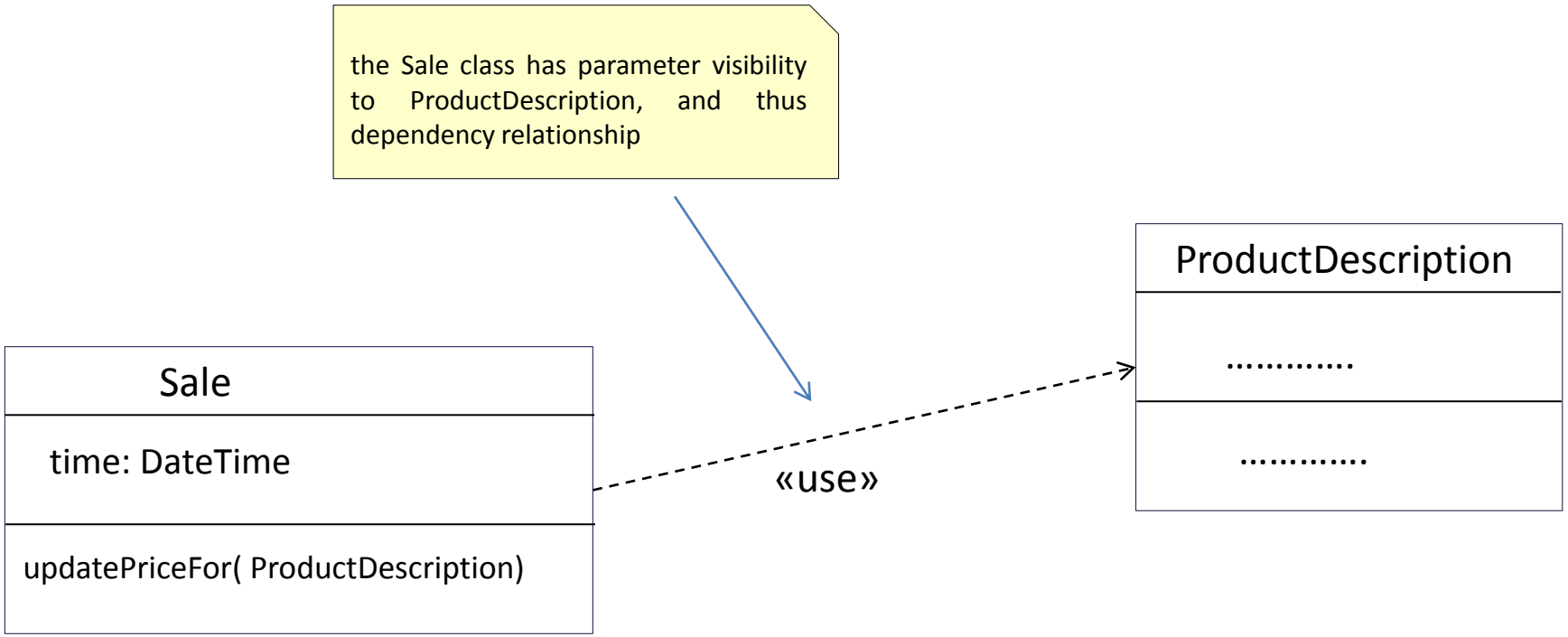
Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Dependencies

- In general, a dependency relationship between two model elements is represented using a dotted arrow, as shown below:

- Various other kinds of dependency relationships exist, and are defined using various available keywords, like:
- «call», «use», «send», «create», etc.

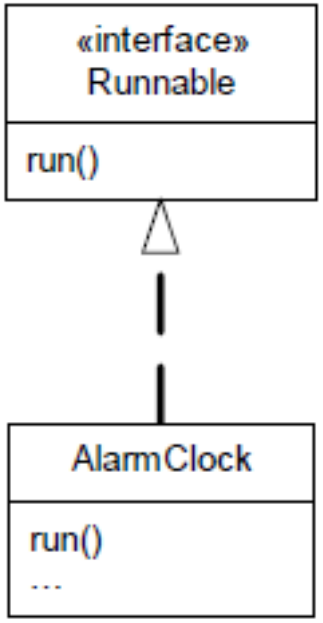
Class Diagrams: Representing Dependencies



Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Interfaces

- «interface» : placed above the classifier name, used to specify that classifier is an interface
- An interface can be realized or implemented by another concrete class. This relationship is represented using realization relationship, as shown below:



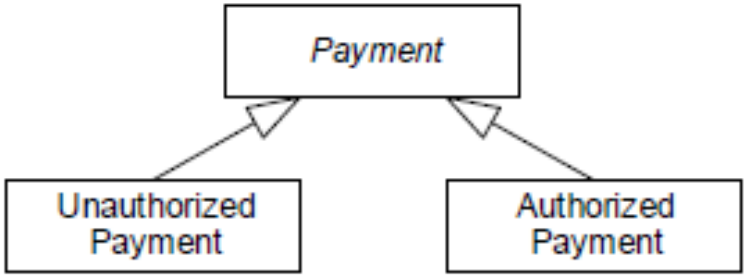
Realization relationship in UML: Notations

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Inheritance

Generalization or Inheritance relationship

- Shown with a solid line and fat triangular arrow from the subclass to super class, as shown below:

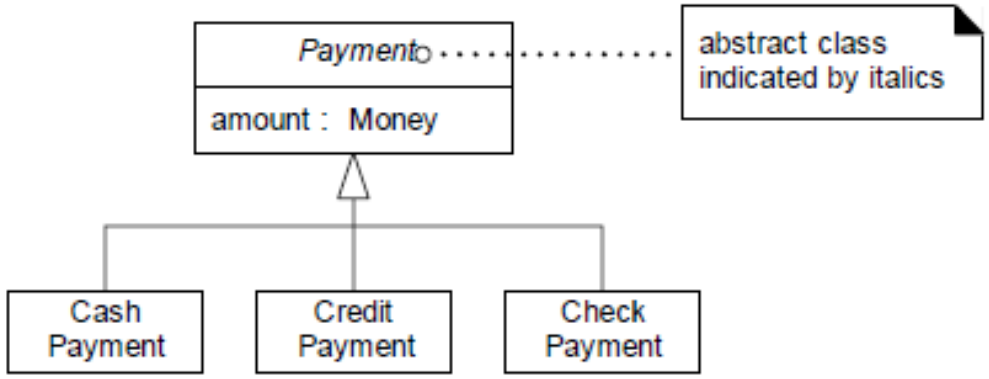


Generalization in UML: Notations

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Abstract Classes

- {abstract}: placed after classifier name or operation name to represent abstraction
- Italicizing the name of the classifier or operation name can also be used to represent abstract classes or operations.



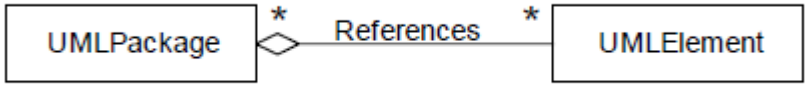
Abstract classes in UML: Notations

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Aggregation

Aggregation:

- It is an association that represents a part-whole relationship.
- It is shown by a hollow-diamond on the end of the path attached to the aggregate class.



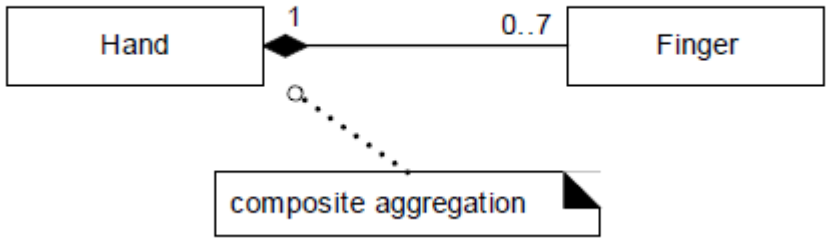
Aggregation in UML: Notations

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Class Diagrams: Representing Composition

Composition:

- It is a stronger form of association in which the composite has responsibility for managing its parts, such as their creation, destruction, allocation or de-allocation.
- It is shown by a filled-diamond on the composite end.
- If the composite is destroyed, its parts must either be destroyed, or attached to another composite- no free floating parts exist.



Composition in UML: Notations

Courtesy: Adapted from Applying UML and Patterns, Craig Larman, 3rd edition

Snakes And Ladders: Partial Class Diagram