Slide3:

[pic1]

Class diagram Describes the structure of the system in terms of classes and objects

A **class** is a [**classifier**](https://www.uml-diagrams.org/classifier.html) which describes a set of objects that share the same

* [**features**](https://www.uml-diagrams.org/uml-core.html#feature)
* [**constraints**](https://www.uml-diagrams.org/constraint.html?context=class-diagrams)
* **semantics (meaning)**.

A class is shown as a solid-outline rectangle containing the class name, and optionally with compartments separated by horizontal lines containing features or other members of the classifier.

[pic2]

shown with three **compartments**,

- **class name**

**- attribute**

**- operation**

[pic3]

Attributes or operations may be grouped by **visibility**. A visibility keyword or symbol in this case can be given once for multiple features with the same visibility.

[pic4]

Attributes or operations may be grouped by **visibility**. A visibility keyword or symbol in this case can be given once for multiple features with the same visibility.

Slide4:

**Visibility** allows to constrain the usage of a **named element**, either in  [**namespaces**](https://www.uml-diagrams.org/namespace.html) or in access to the element. It is used with [**classes**](https://www.uml-diagrams.org/class.html), [**packages**](https://www.uml-diagrams.org/package-diagrams.html#package), [**generalizations**](https://www.uml-diagrams.org/generalization.html?context=class-diagrams), [**element import**](https://www.uml-diagrams.org/element-import.html), [**package import**](https://www.uml-diagrams.org/package-import.html).

UML has the following types of **visibility**:

* public (+)
* package (~)
* protected (#)
* private (-)
* underline static.

If a **named element** is not owned by any [**namespace**](https://www.uml-diagrams.org/namespace.html), then it does not have a visibility.

Slide6: Association

|  |  |
| --- | --- |
| Indicator | Meaning |
| 0..1 | Zero or one |
| 1 | One only |
| 0..\* | Zero or more |
| 1..\* | One or more |
| n | Only n (where n > 1) |
| 0..n | Zero to n (where n > 1) |
| 1..n | One to n (where n > 1) |

Slide 7: Inheritance

A subclass assumes attributes and operations of the superclass

A **Generalization** is shown as a line with a hollow triangle as an arrowhead between the symbols representing the involved classifiers. The arrowhead points to the symbol representing the general classifier. This notation is referred to as the **"separate target style."**

Multiple Generalization relationships that reference the same general classifier can also be connected together in the **"shared target style."**

Slide 8:

In cases where interfaces are represented using the rectangle notation, **interface realization** dependency is denoted with interface realization arrow. The classifier at the tail of the arrow implements the interface at the head of the arrow.

Slide9:

***Generalization***is the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass. Shared characteristics can be attributes, associations, or methods.

***specialization***means creating new subclasses from an existing class. If it turns out that certain attributes, associations, or methods only apply to some of the objects of the class, a subclass can be created.

<https://sourcemaking.com/uml/modeling-it-systems/structural-view/generalization-specialization-and-inheritance>

Slide 10:

[**Dependency**](https://www.uml-diagrams.org/dependency.html) relationship is used on class diagrams to show [**usage**](https://www.uml-diagrams.org/dependency.html#usage) dependency or [**abstraction**](https://www.uml-diagrams.org/abstraction.html?context=class-diagrams).

A dependency is generally shown as a dashed arrow between two model elements. The model element at the tail of the arrow (the **client**) **depends on** the model element at the arrowhead (the **supplier**). The arrow may be labeled with an optional stereotype and an optional name.

Class A depends upon class B means that if class B is changed, class A must also be affected

Slide 11:

**Aggregation** (aka **shared aggregation**) is shown as binary association decorated with a **hollow diamond** as a terminal adornment at the aggregate end of the association line.

“belongs to” relationship , Aggregation indicates structural inclusion of objects of one class by another class

Slide 12: Composition

Sometimes an object is made up of other objects. For example, an airplane is made up of a fuselage, wings, engines, landing gear, flaps, and so on. Figure 8 presents an example using composition, modeling the fact that a building is composed of one or more rooms, and then, in turn, that a room may be composed of several subrooms (you can have recursive composition). In UML 2, aggregation would be shown with an open diamond.

* “part-of” relaHonship
* •  Composed object is created/destroyed when composing

object is created/destroyed  
• Composed instance is not shared by other classes

<https://www.uml-diagrams.org/class-reference.html>