Packages

- A package is a general-purpose mechanism for organizing elements into groups. Graphically, a package is rendered as a tabbed folder.
- You can group model elements in packages for the following reasons:
  - Organize the model elements so that the model is easier to understand
  - Model the architecture of your system by using packages to represent the various layers or subsystems
- Every package must have a name that distinguishes it from other packages.

Owned Elements

- **A package may own other elements**, including classes, interfaces, components, nodes, collaborations, use cases, diagrams, and even other packages
  - Owning is a composite relationship, which means that the element is declared in the package.
  - If the package is destroyed, the element is destroyed.
  - Every element is uniquely owned by exactly one package.

Owned Elements

- **A package forms a namespace,** which means that elements of the same kind must be named uniquely within the context of its enclosing package.
- **Different kinds of elements may have the same name within a package.**
- **Packages may own other packages.**

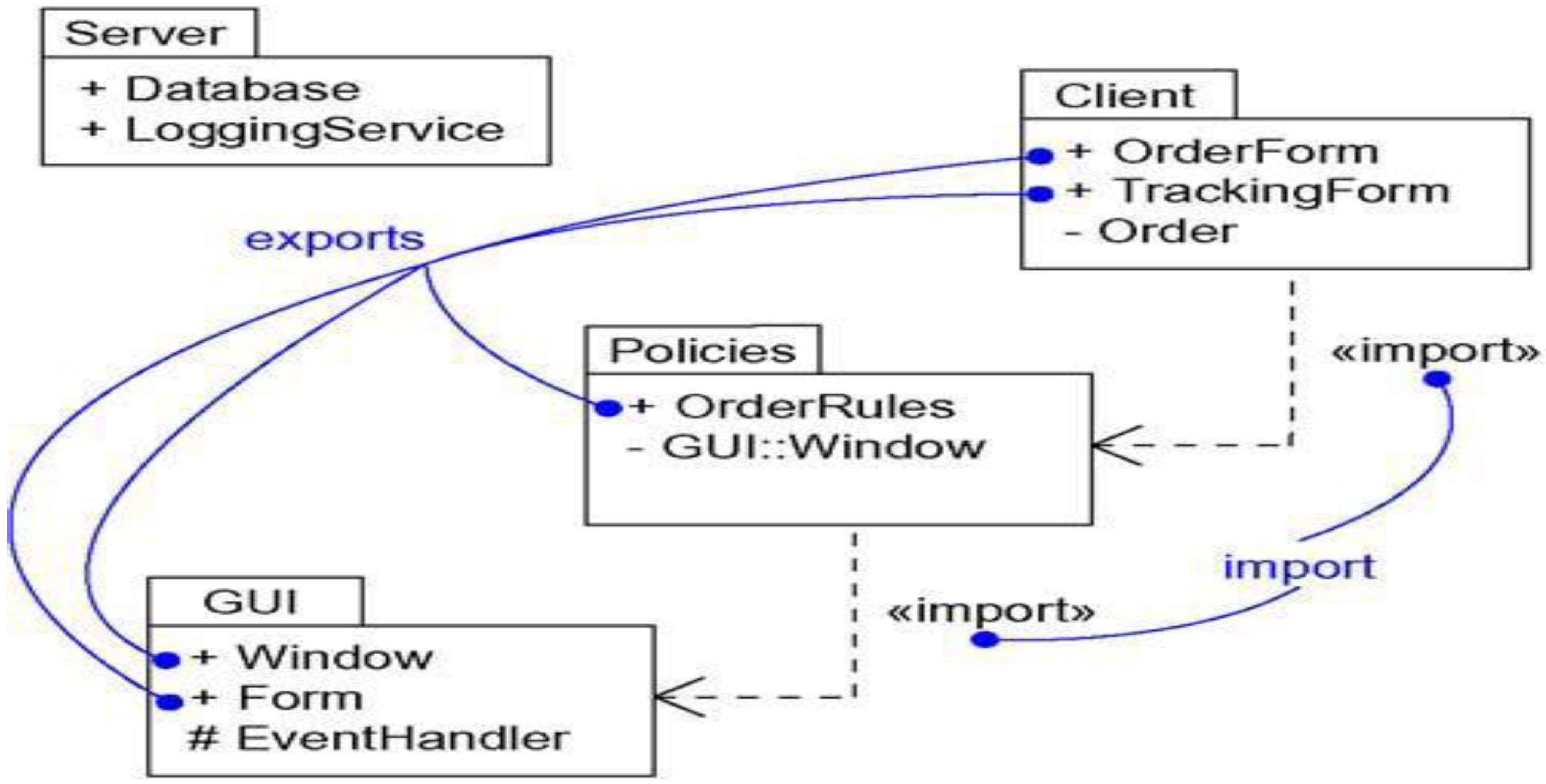
Visibility

- Like a class, the package also have visibility control of the elements
- An element owned by a package is **public(+)**, which means that it is visible to the contents of any package that imports the element's enclosing package.
- **Protected(#)** elements can only be seen by children
- **Private(-)** elements cannot be seen outside the package in which they are declared.

Importing and Exporting

- A in one package and B in another package, both packages sitting side by side.
- A and B are both declared as public parts of their respective packages.
- If A's package imports B's package, A can now see B, although B cannot see A.
- **Importing grants a one-way permission** for the elements in one package to access the elements in another package.
- In the UML, an import relationship as a dependency adorned with the stereotype import.
- The **public parts of a package are called its exports**.

Importing and Exporting

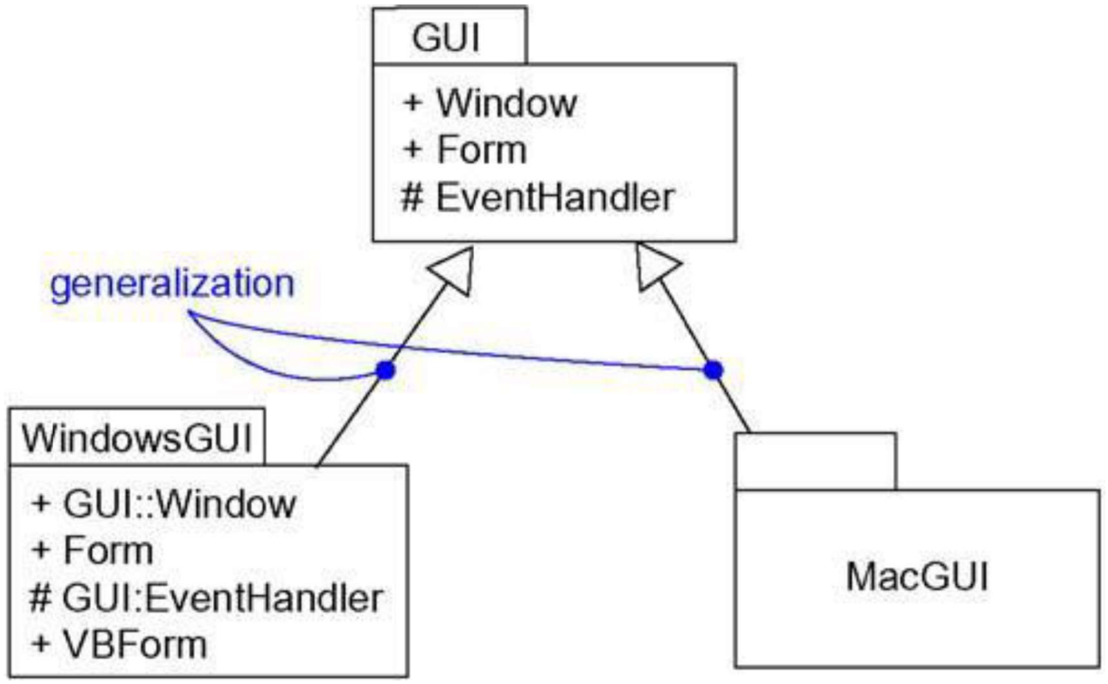


Generalization

- There are two kinds of relationships between packages:
  - Import
    - If element import is **public**, the imported element will be added to the namespace and made visible outside the package
      - Keyword **«import»** indicates **public** element import.
  - Access
    - If element import is **private**, the imported element will be added to the namespace but will not be visible outside the namespace.
      - Keyword **«access»** indicates **private** element import.

Generalization example

- The package GUI is shown to export two classes (Window and Form) and one protected class (EventHandler)
- The package WindowsGUI inherits from GUI, so it includes the classes GUI::Window and GUI::EventHandler.
- In addition, WindowsGUI overrides one class (Form) and adds a new one (VBForm).



Instances

- An instance is a concrete manifestation of an abstraction to which a set of operations can be applied and which has a state that stores the effects of the operations.
- Graphically, an instance is rendered by underlining its name.

Abstractions and Instances

- Most instances you'll model with the UML will be instances of classes although you can have instances of other things, such as components, nodes, use cases, and associations
- In the UML, an instance is easily distinguishable from an abstraction. To indicate an instance, you underline its name.