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

Visibility

Visibility

- Visibility is the ability of one object to see or have reference to another object.
- There are four common ways in which visibility can be achieved from object A to object B:
  - Attribute visibility: B is an attribute of A.
  - Parameter visibility: B is a parameter of a method of A.
  - Local visibility: B is a (non-parameter) local object in a method of A.
  - Global visibility: B is in some way globally visible.

Attribute Visibility

innovate achieve lead

- Attribute visibility from A to B exists when B is an attribute of A.
- Visibility exists as long as both A and B exists.

Example:

```
public class Register
{
…
private ProductCatalog catalog;
…
…
}
```

Parameter Visibility

- Parameter visibility from A to B exists when B is passed as a parameter to a method A.
- Visibility persists only within the scope of the method.

Example:

```
makeLineItem (ProductDescription desc, int qty)
{
…
sl = new SalesLineItem(desc, qty);
…
…
}
```

Local Visibility

innovate achieve lead

- Local visibility from A to B exists when B is declared as a local object within a method of A.
- Visibility persists only within the scope of the method.

Example:

```
makeLineItem (ProductDescription desc, int qty)
{
…
ProductDescription description = desc;
…
…
}
```

Global Visibility

- Global visibility from A to B exists when B is global to A.
- Visibility exists as long as both A and B exists.
- In Java, Singleton pattern is used to achieve global visibility.

Package Diagram

Package

- Package is a **namespace** used to group together elements that are semantically related to each other.
- Usually, packages are used to group classes that are related to each other.
- In Java, directory structure are also used to represent grouping of classes into packages.

- If a package is removed from a model, all the elements **owned** by the package will be removed.
- **Owned members** of a package should all be **packageable elements**.
- Package by itself is also a packageable element.
- Any package could also be a member of other packages.

Need for Packages

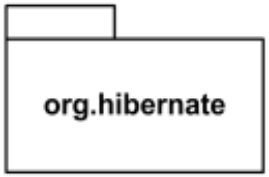
- To logically organize classes or code.
- Segregate classes implementing related functionality into a separate bundle so that they can easily be used and modified.
- To increase the modularity of the system.
- To depict a high-level overview of your system design or architecture.
- To depict a high-level overview of your system requirements (by organizing use cases into packages).

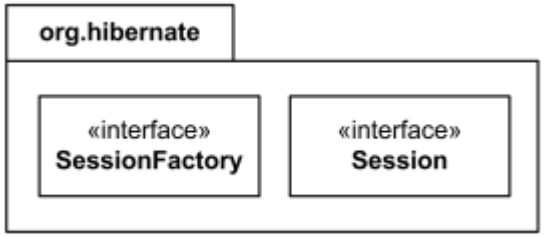
Packages

- Because package is a namespace, elements of related or the same type should have unique names within the enclosing package.
- Elements belonging to different packages can have same names.
- A package can import either individual members of other packages or all the members of other packages.

Package Diagram: Notations

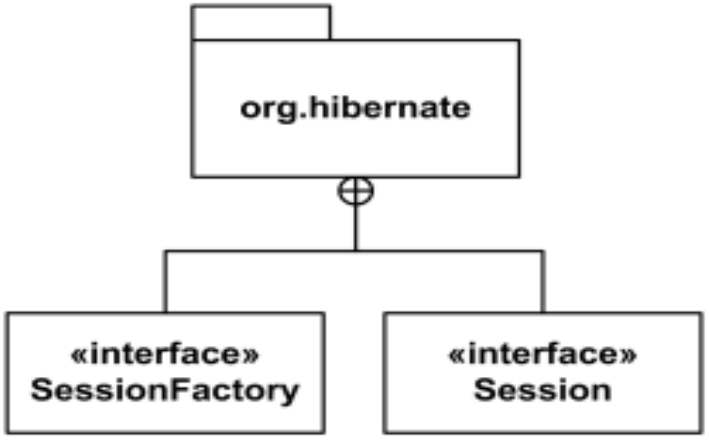
- A package is rendered as a tabbed folder – a rectangle with a small tab attached to the left side of the top of the rectangle.
- If the members of the package are not shown inside the package rectangle, then the name of the package should be placed inside.
- If the members of the package may be shown within the boundaries of the package, then the name of the package should be placed on the tab.





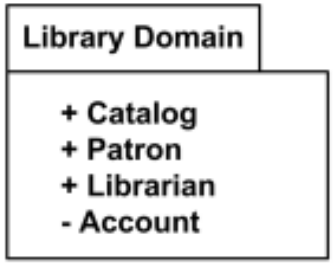
Package Diagram: Notations

- Members of the package may be shown **outside** of the package by branching lines from the package to the members.
- A **plus sign (+) within a circle** is drawn at the end attached to the namespace (package).
- This notation for packages is semantically equivalent to **composition** (which is shown using solid diamond.)



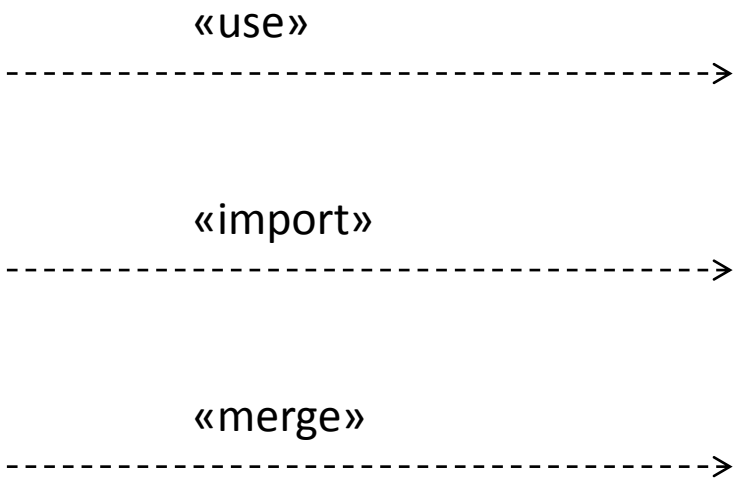
Package Diagram: Notations

- Elements can be referred to within a package using **non-qualified** names.
- If an element that is owned by a package has visibility, it could be only public or private visibility.
- Protected or package visibility is not allowed.
- The visibility of a package element may be indicated by preceding the name of the element by a visibility symbol ("+" for public and "-" for private).
- The public elements of a package are always accessible outside the package through the use of qualified names.



Package Diagram: Notations

- Dependencies between packages can be shown using dotted arrows.



Package Diagram: Guidelines

- Package logically related classes or program elements together.
- Place the classes of a framework in the same package.
- Classes in the same inheritance hierarchy typically belong to the same package.
- Classes related to one another via aggregation or composition often belong in the same package.
- Classes having high coupling (interaction) often belong to the same package.

Plan ahead…..

Go through Lecture Videos:

- Module 5.1: Visibility
- Module 5.3: Package Diagram

Agenda: Lecture 8

- Design Patterns (GOF)