



MUST
Wisdom & Virtue

MIRPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF SOFTWARE ENGINEERING

Software Design & Architecture

(Lecture # 5)

*Saba Zafar
(Lecturer)*

LECTURE CONTENTS

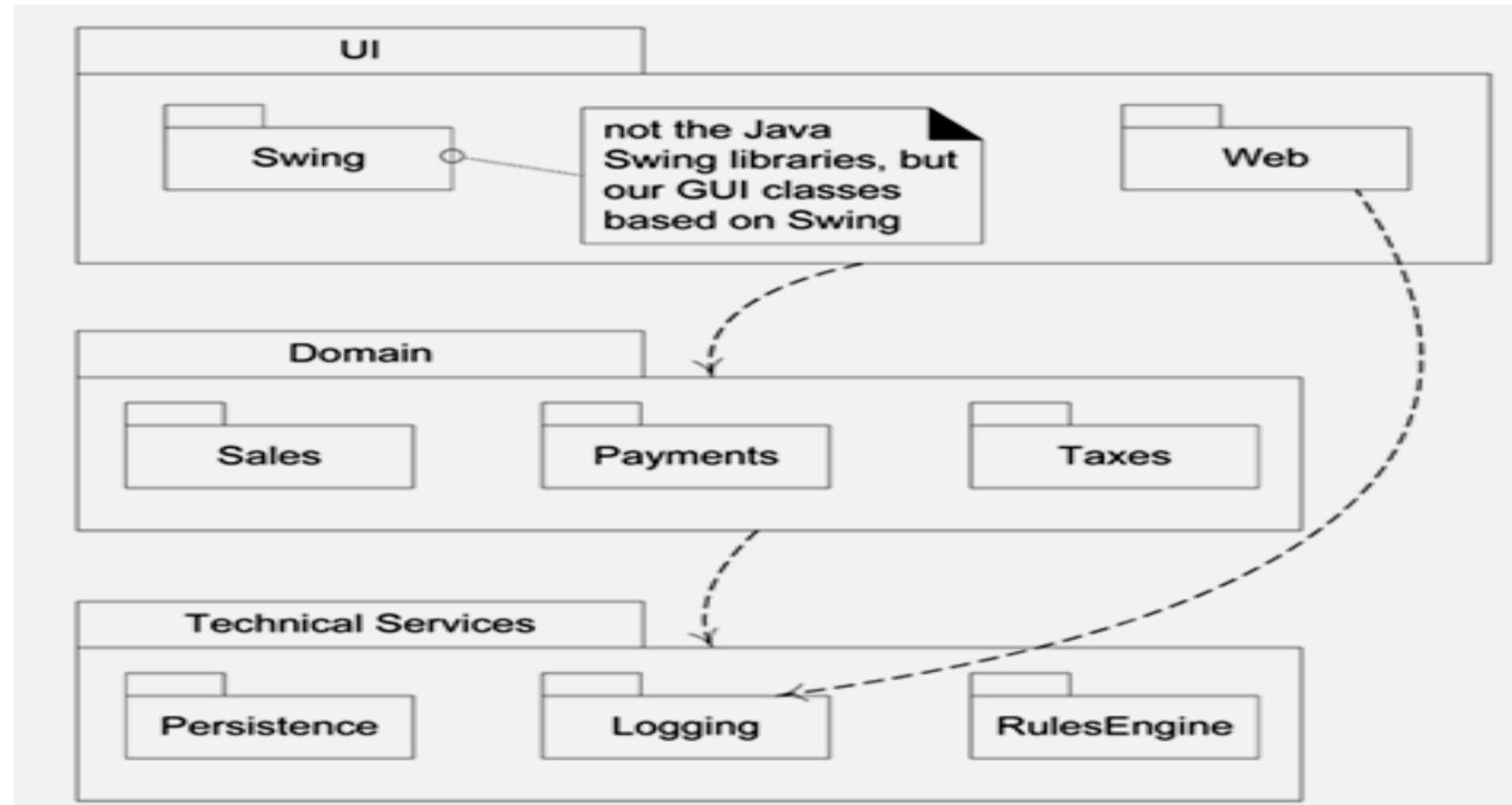
1. UML Package Diagram
2. Ownership and References In Package Diagram
3. Classes in Package Diagram
4. Use Cases in Package Diagram
5. Examples of Package Diagram



UML PACKAGE DIAGRAM

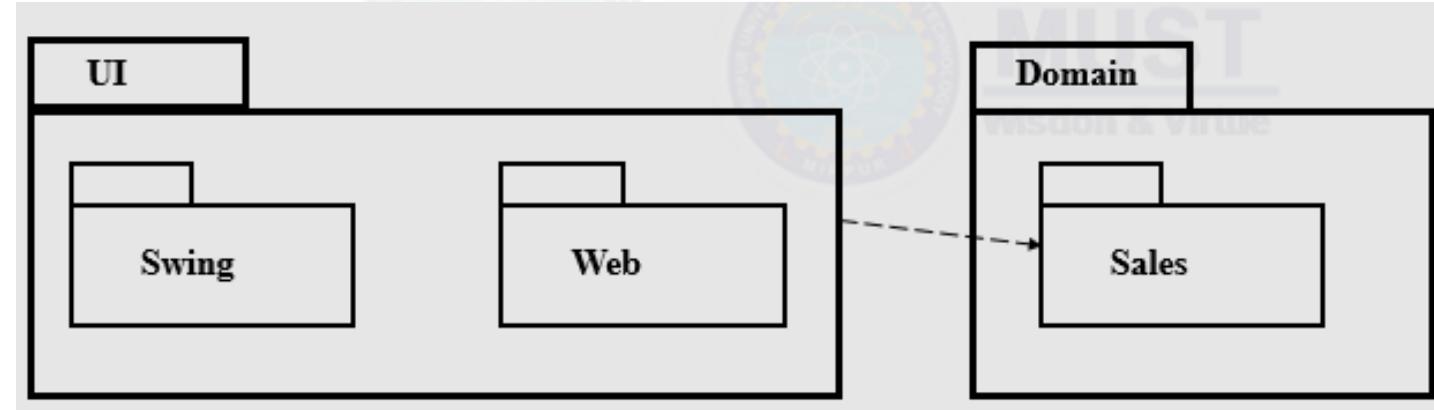
- UML Package Diagrams provide a way to **group elements**
- It can be used to **simplify** complex class diagrams, it can group classes into packages.
- A package is a collection of logically related UML elements.
- A UML package can group anything, e.g., classes, use-cases, other packages, etc.
- Packages as components, can be **nested** inside other packages
- Dependencies among different packages can be shown with the **dependency line**

PACKAGE DIAGRAM



PACKAGE DIAGRAM

- UML package is shown as a **tabbed folder**
- Subordinate packages may be shown within it
- The package name is within the tab if the package **depicts** its elements; otherwise, it is centered within the folder itself



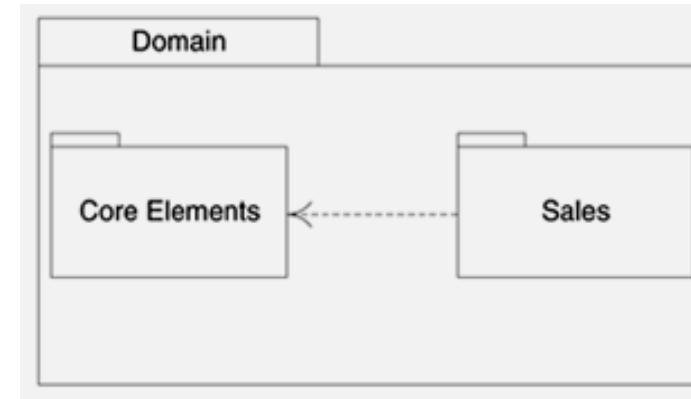
OWNERSHIP AND REFERENCES IN PACKAGE DIAGRAM

- An element is **owned** by the package within which it is defined, but may be **referenced** in other packages
- In that case, the element name is qualified by the package name using the pathname format **PackageName::ElementName**
- A class shown in a **foreign package** may be modified with new associations, but must otherwise remain unchanged



PACKAGE DEPENDENCIES

- If a model element is in some way **dependent** on another, the dependency may be shown with a **dependency relationship**, depicted with an **dotted arrowed line**
- A package dependency indicates that elements of the dependent package in some way **know about** or **are coupled to** elements in the target package
- **For example**, if a package references an element owned by another, a dependency exists. Thus, the **Sales package** has a dependency on the **Core Elements** package



CLASSES IN A PACKAGE

- Following types of classes can typically be grouped together in one package:
 - a. Classes in the same **inheritance hierarchy**
 - b. Classes related to one another via **composition**
 - c. Classes that **collaborate** with each other (a lot of information reflected by your **sequence diagrams** and **communication diagrams**)

USE-CASES IN A PACKAGE

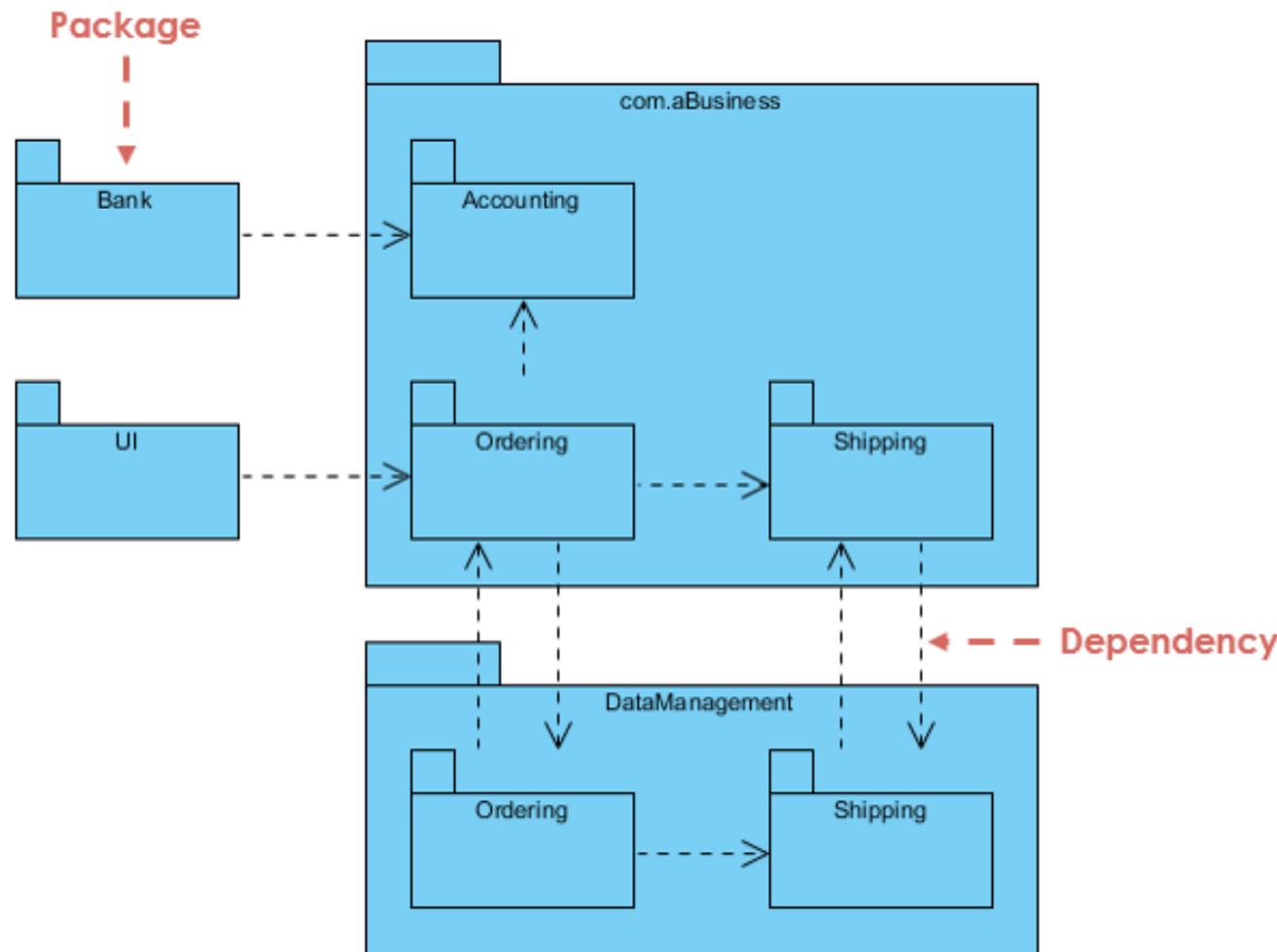
- Following categories of use-cases are typically grouped together in a package:
 - Included and extending use cases belong in the same package as the base/parent use case
 - The use cases with which the main actors are involved



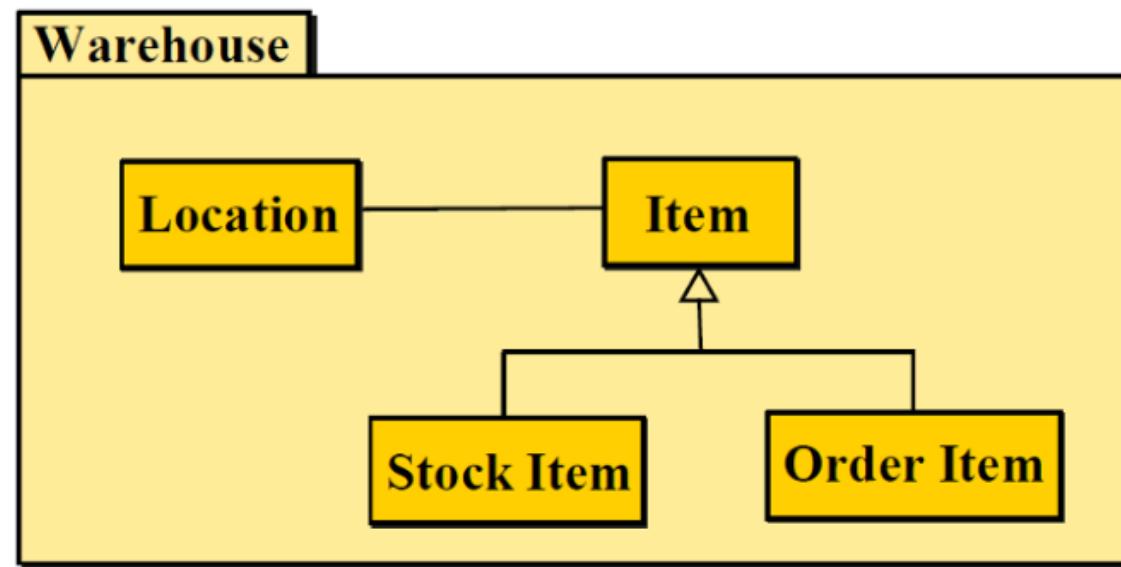
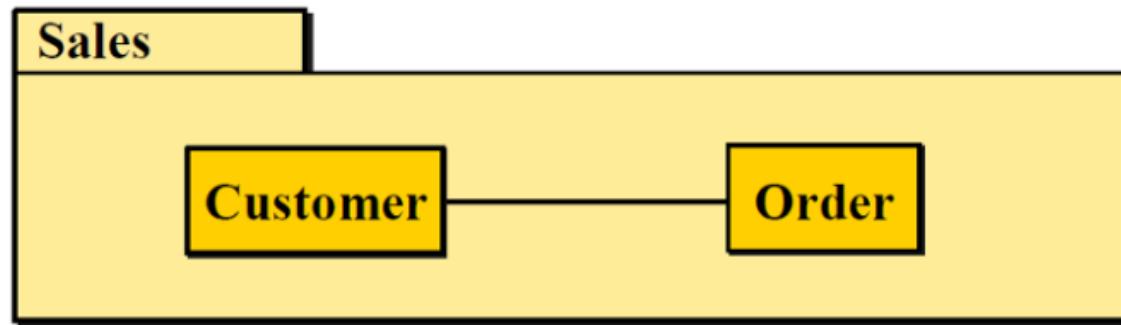
REMEMBER!!!

- Packages should be cohesive
- Anything you put into the package should make sense when considered with the rest of the contents of the package
- To determine whether a package is cohesive, a good test is you should be able to give your package a short, descriptive name

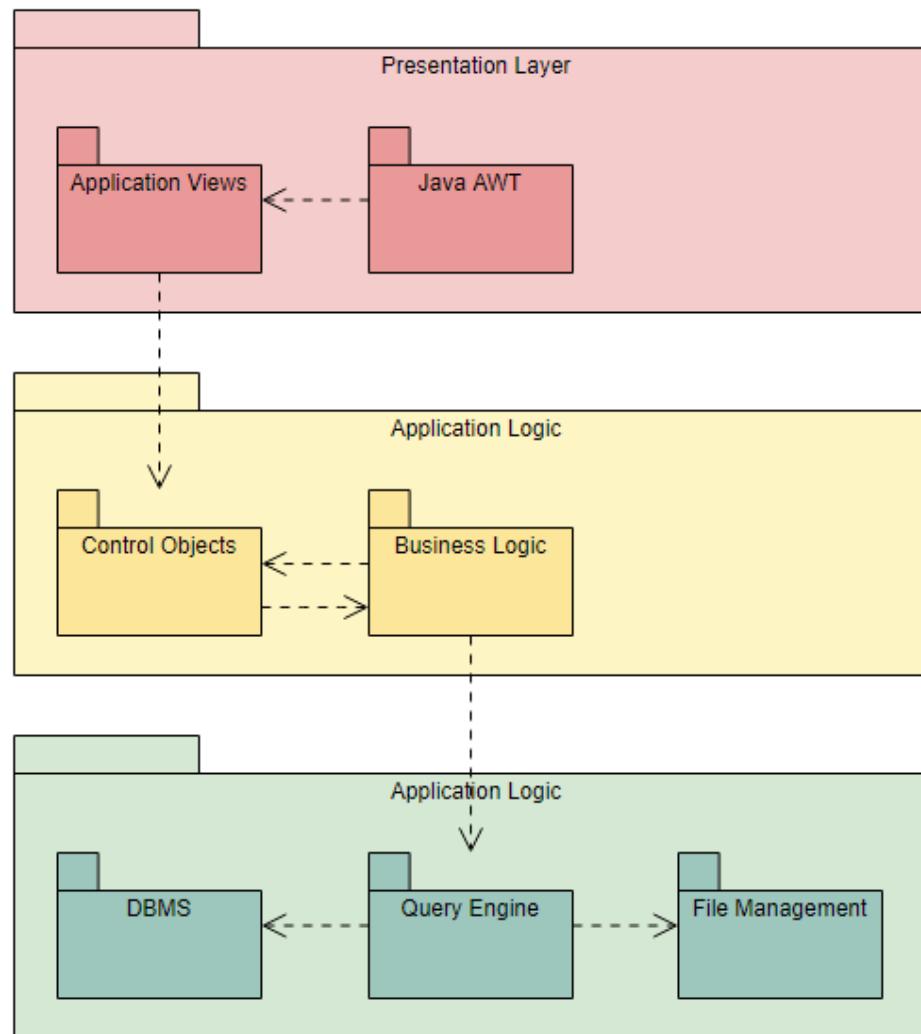
Example 1



Example 2

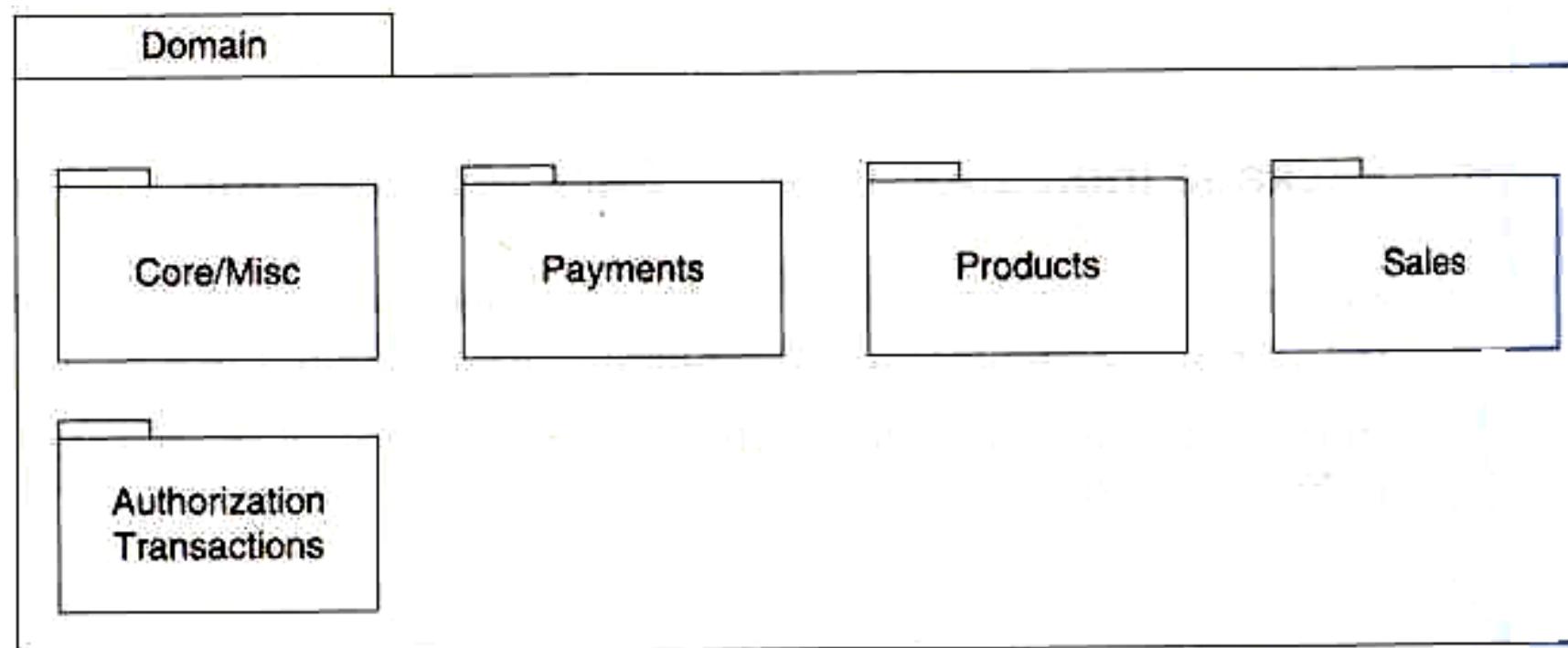


Example 3 – Model View Controller (MVC) Structure



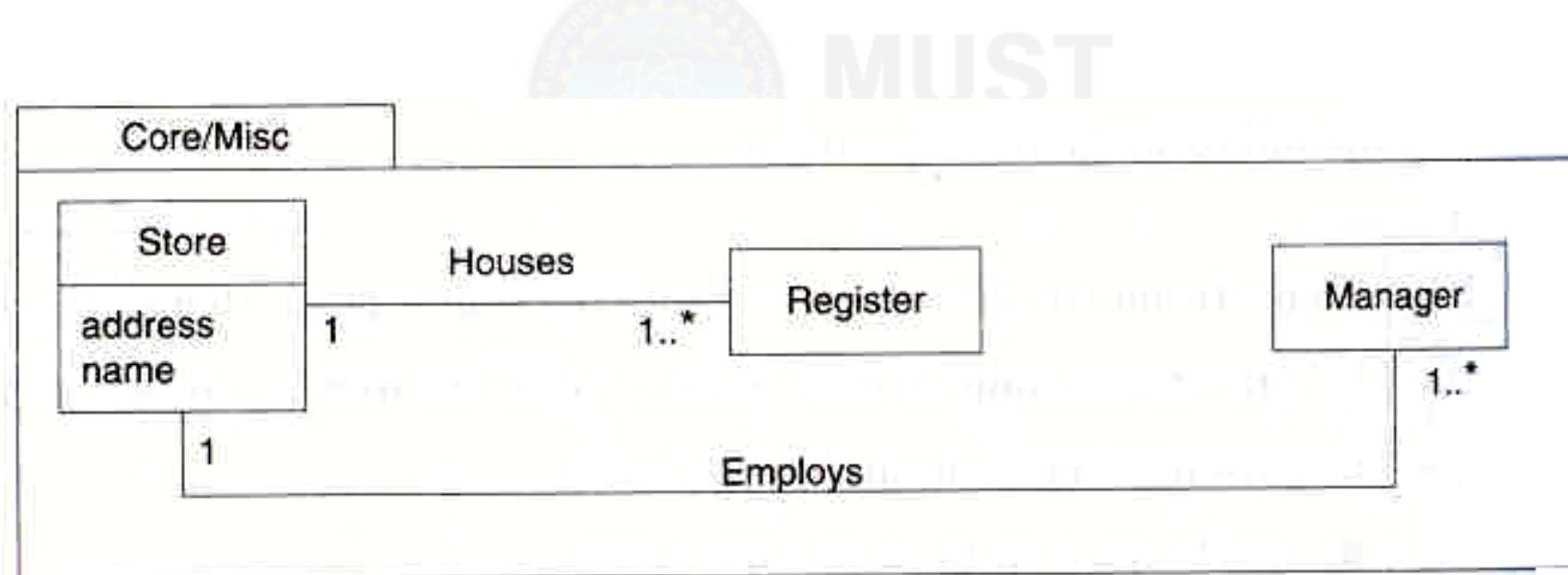
Example 4: POS PACKAGES

Showing the Domain layer :

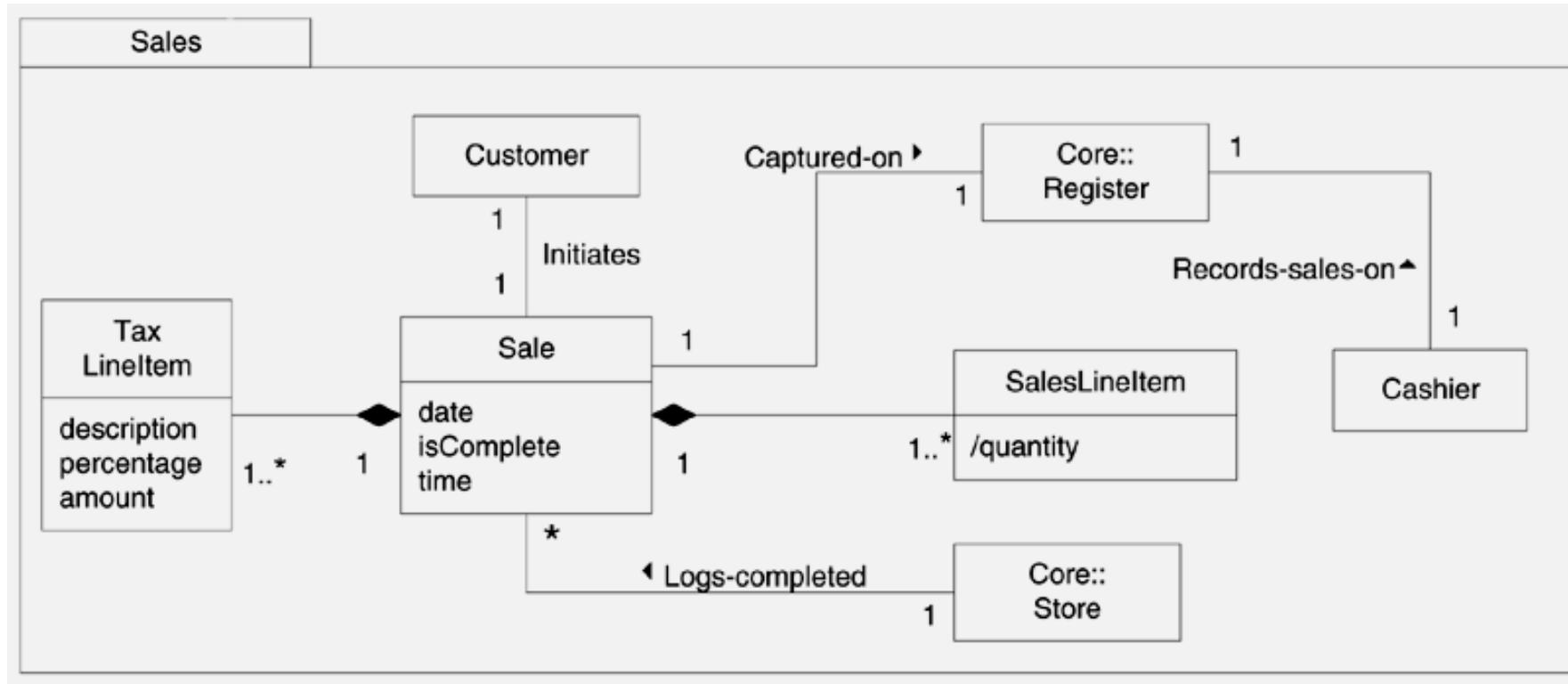


POS: CORE/MISC PACKAGE

- A Core/Misc package is useful to own widely shared concepts or those without an obvious home



POS: SALES PACKAGE



THANKS