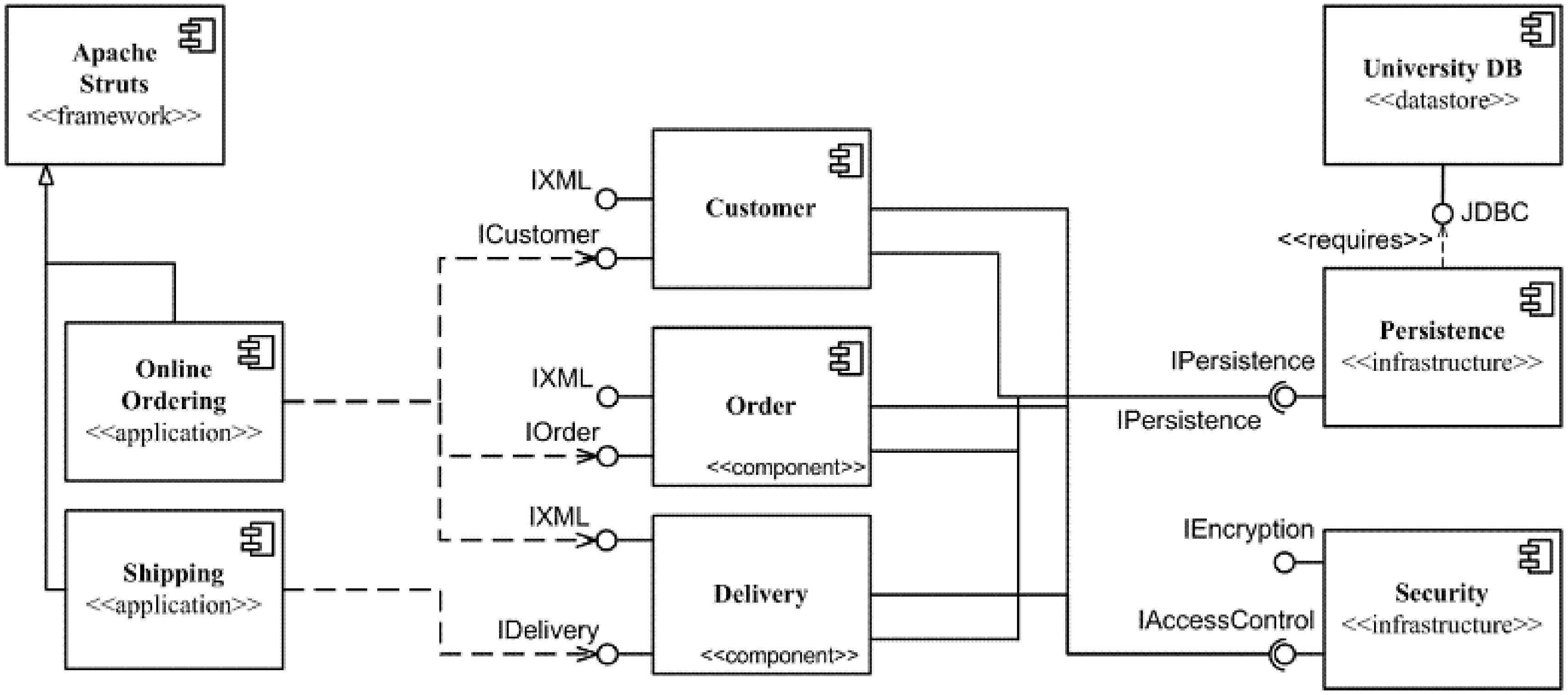
8. Static Modeling using the Unified Modeling Language (UML) - **Component Diagram**

Component Diagram

UML component diagrams shows the dependencies among software components, including the classifiers that specify them (for example implementation classes) and the artifacts that implement them; such as source code files, binary code files, executable files, scripts and tables.

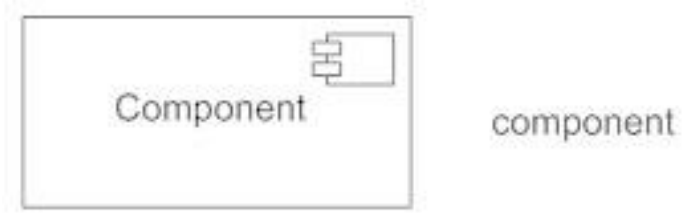
Component Diagram



Component Diagram

Component

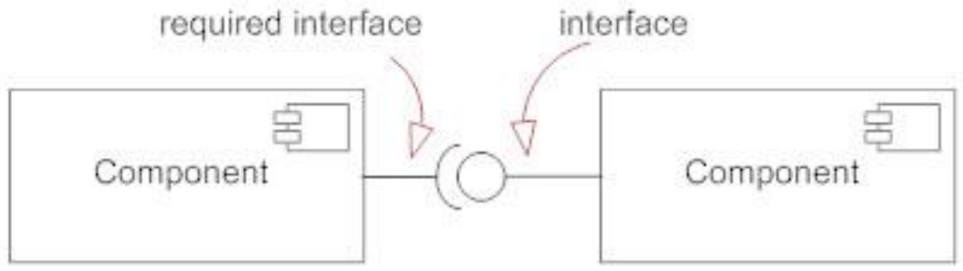
A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



Component Diagram

Interface

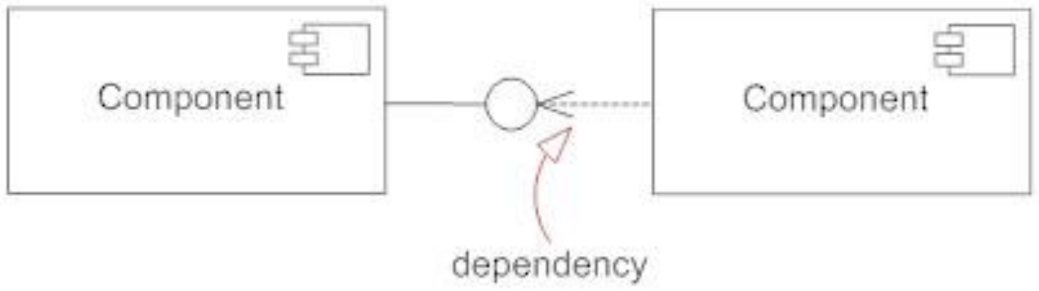
An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.



Component Diagram

Dependencies

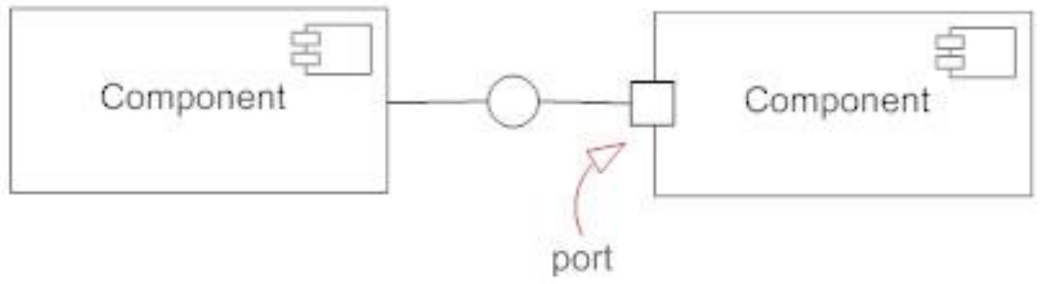
Draw dependencies among components using dashed arrows.



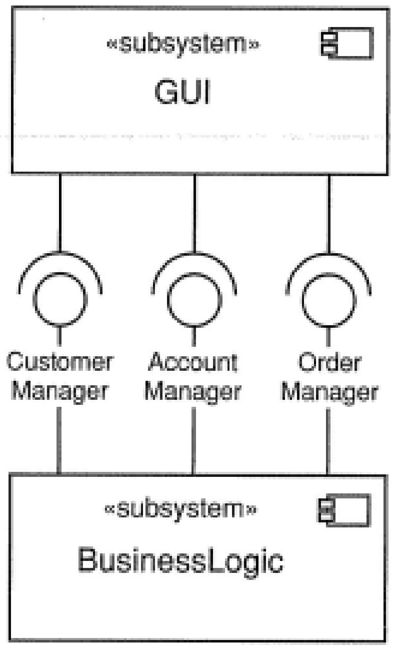
Component Diagram

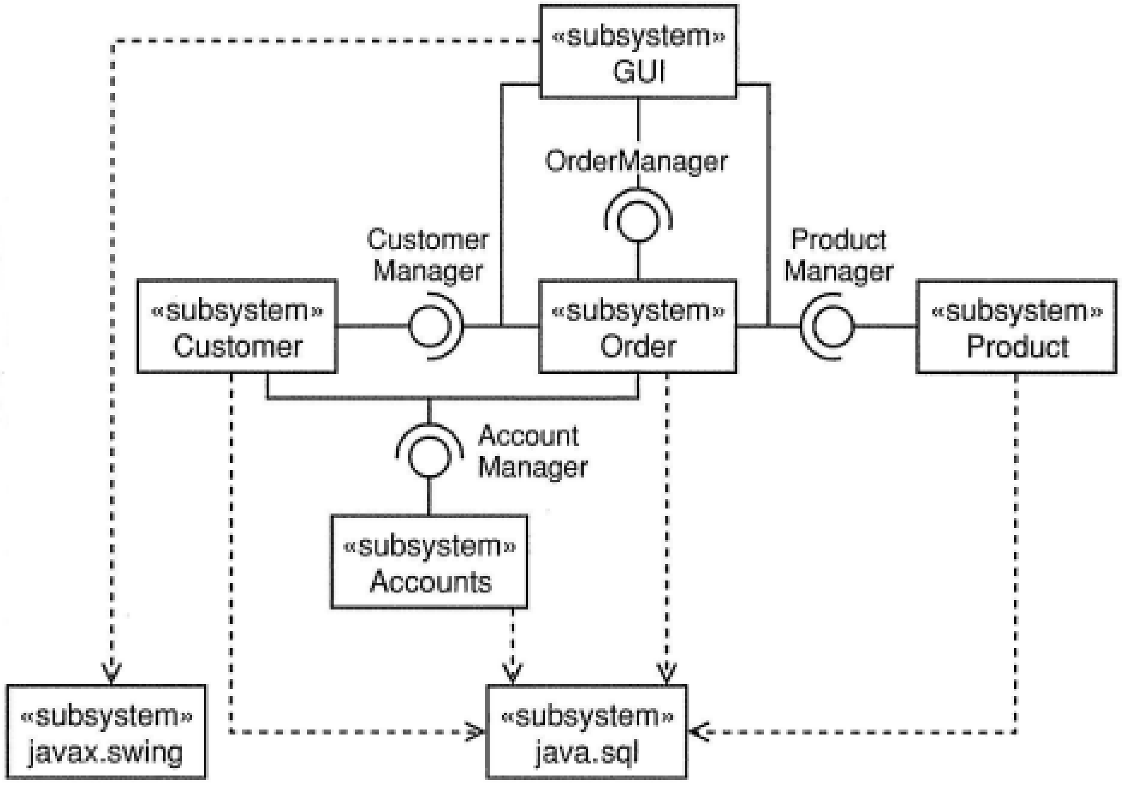
Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.

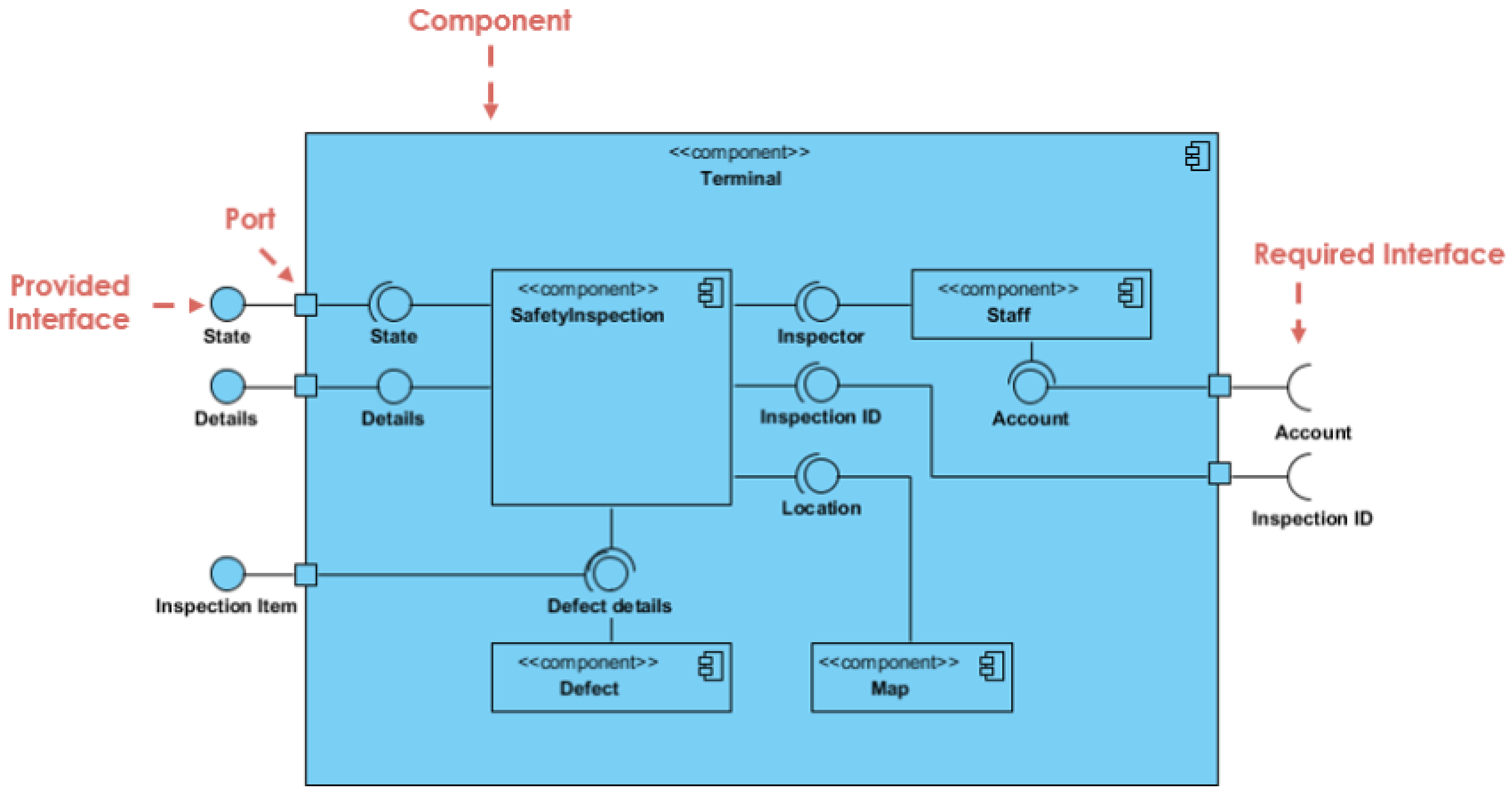


Component Diagram



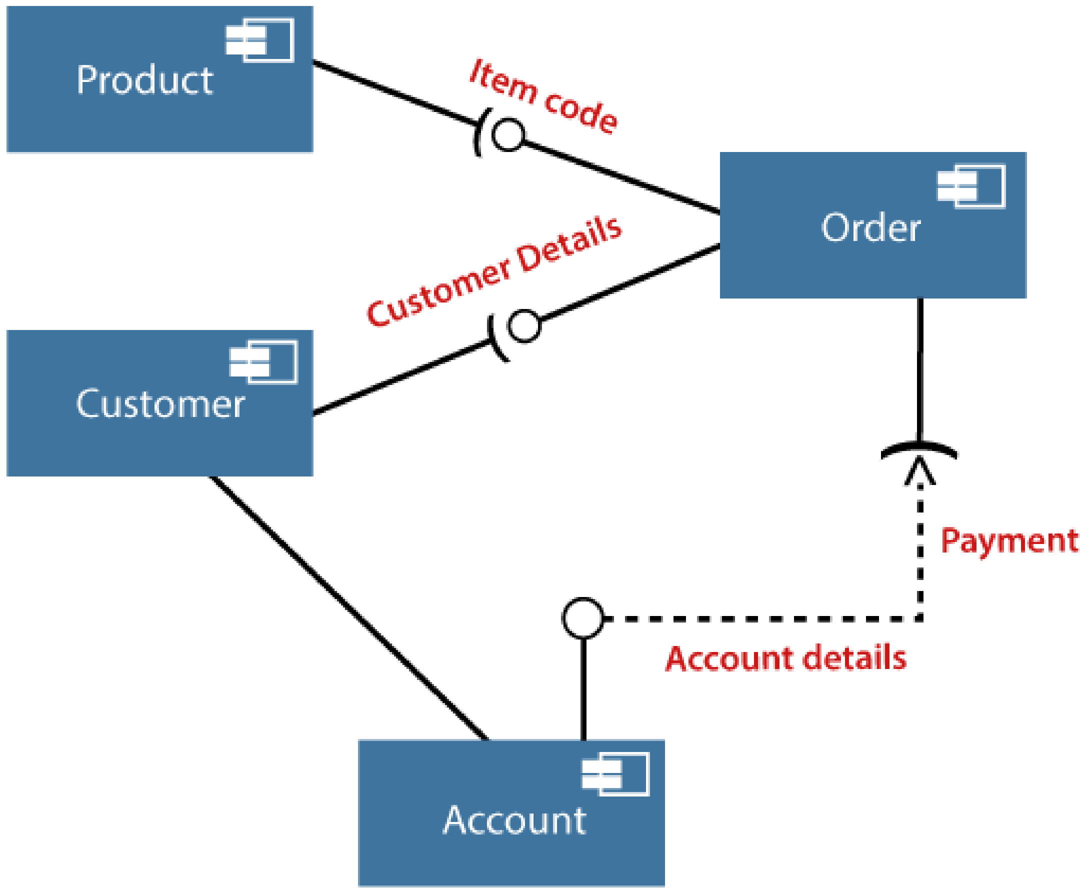


Component Diagram



Component Diagram

A component diagram for an online shopping system is given below:



Component Diagram

The component diagram is a special purpose diagram, which is used to visualize the static implementation view of a system. It represents the physical components of a system, or we can say it portrays the organization of the components inside a system. The components, such as libraries, files, executables, etc. are first needed to be organized before the implementation.

8. Static Modeling using the Unified Modeling Language (UML) - **Package Diagram**

Package Diagram

- Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages.

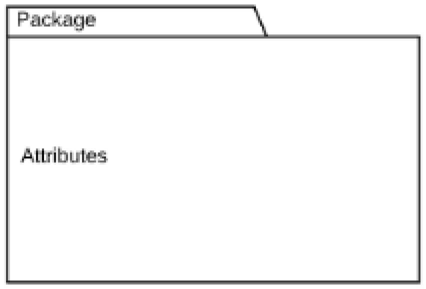
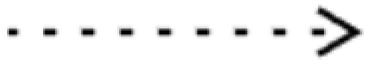
- A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram.

Package Diagram

- Package diagrams are most commonly used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

Package Diagram

Basic components of a package diagram

Symbol Image	Symbol Name	Description
	Package	Groups common elements based on data, behavior, or user interaction
	Dependency	Depicts the relationship between one element (package, named element, etc) and another

Package Diagram

Here are the basic components within a package diagram:

Package: A namespace used to group together logically related elements within a system. Each element contained within the package should be a packageable element and have a unique name.

Package Diagram

Packageable element: A named element, possibly owned directly by a package. These can include events, components, use cases, and packages themselves. Packageable elements can also be rendered as a rectangle within a package, labeled with the appropriate name.

Package Diagram

Dependencies: A visual representation of how one element (or set of elements) depends on or influences another. Dependencies are divided into two groups: access and import dependencies.

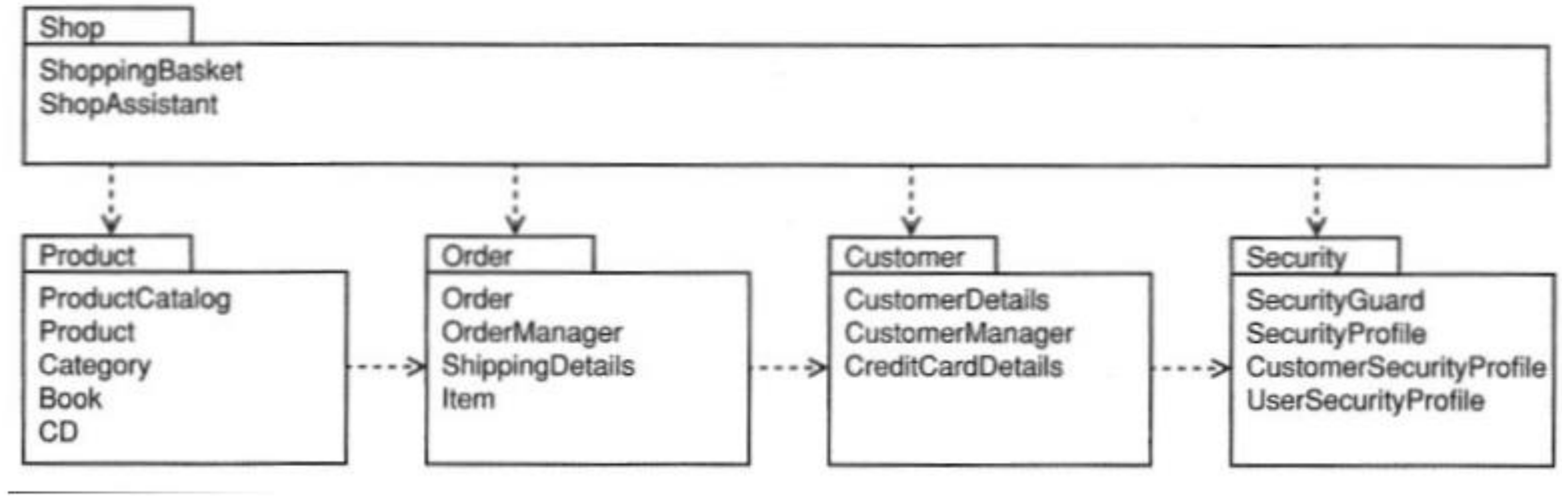
Element import: A directed relationship between an importing namespace and an imported packageable element. This is used to import select individual elements without resorting to a package import and without making it public within the namespace.

Package Diagram

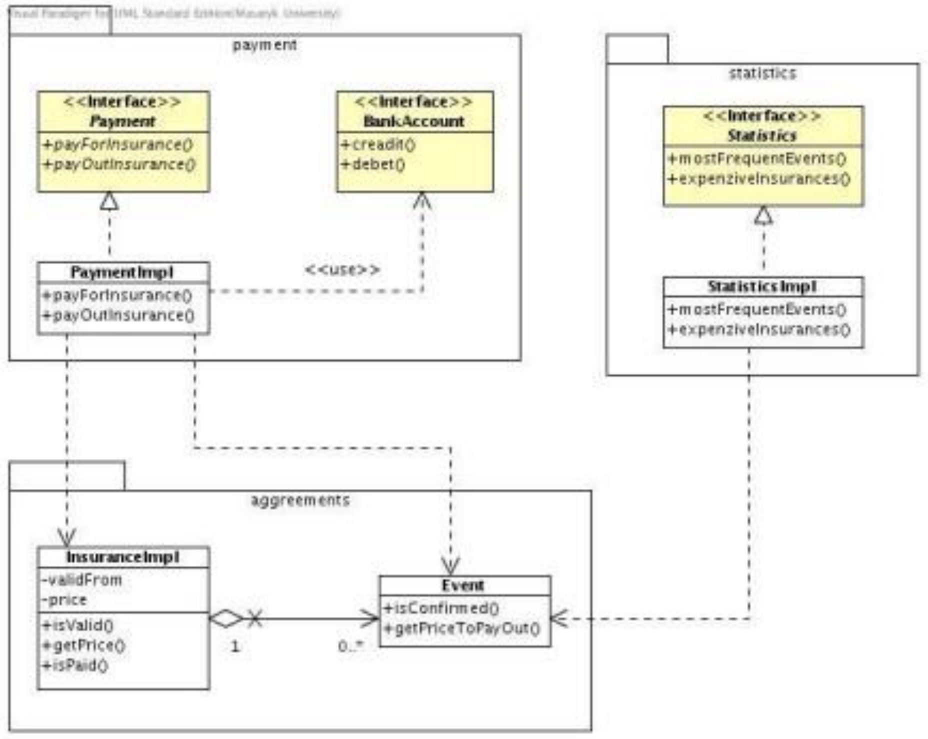
Package import: A directed relationship between and importing namespace and an imported package. This type of directed relationship adds the names of the members of the imported package to its own namespace

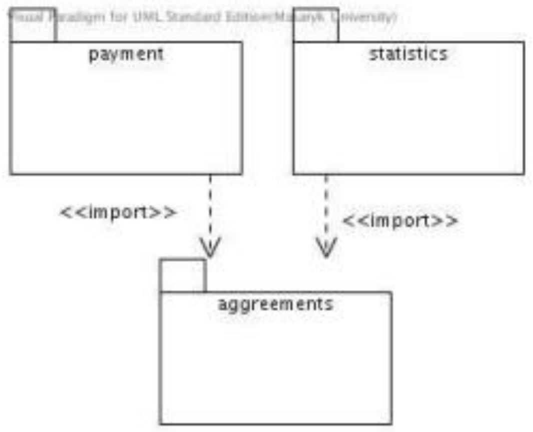
Package merge: A directed relationship in which the contents of one package are extended by the contents of another. Essentially, the content of two packages are combined to produce a new package.

Package Diagram



Package Diagram





Component Vs Package Diagram

- The difference between package diagrams and component diagrams is that Component Diagrams offer a more semantically rich grouping mechanism. With component diagrams all of the model elements are private, whereas package diagrams only display public items.

Component Vs Package Diagram

- Component diagram shows an encapsulated class and its interfaces, ports and internal structure consisting of nested components and connectors.
- It addresses the static design implementation view of a system.
- Package diagram shows the decomposition of model itself into organizational units and their dependencies.