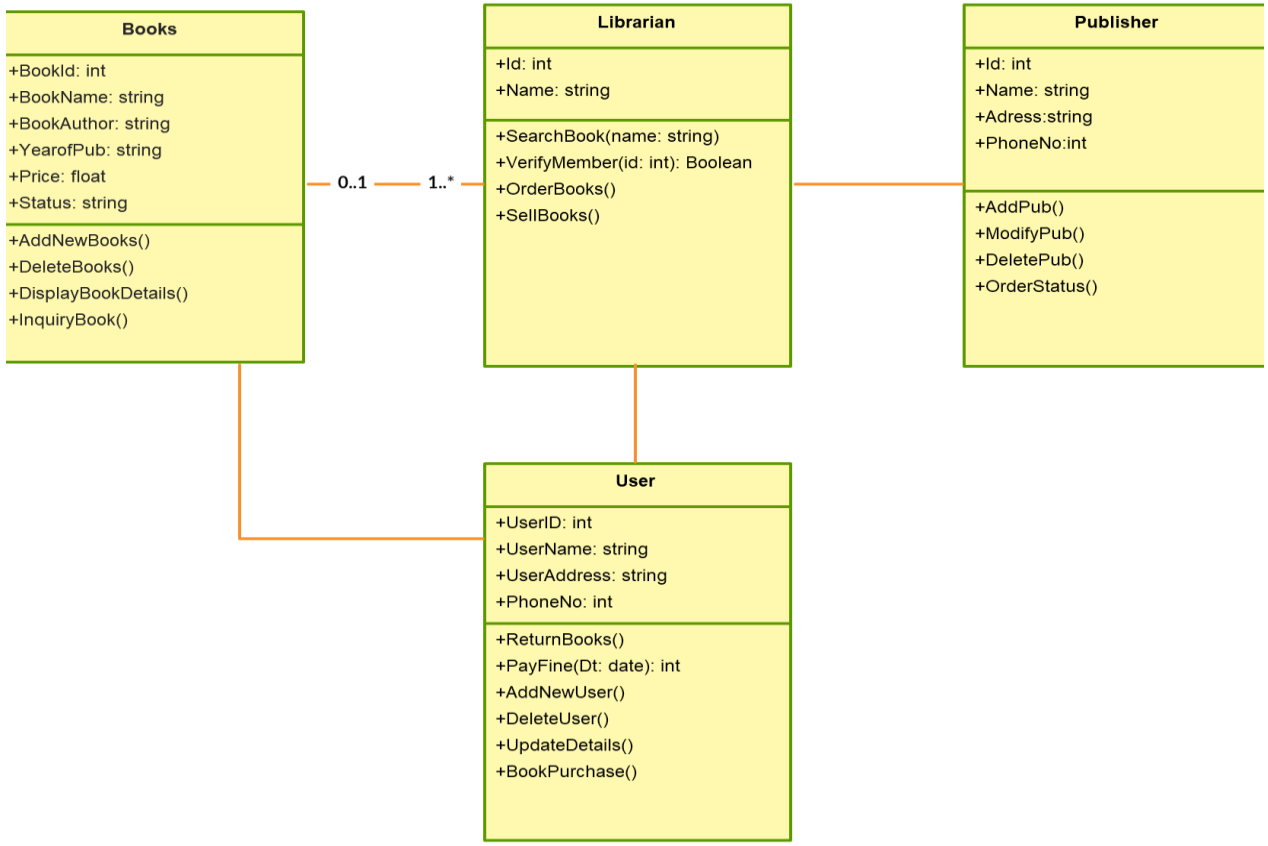
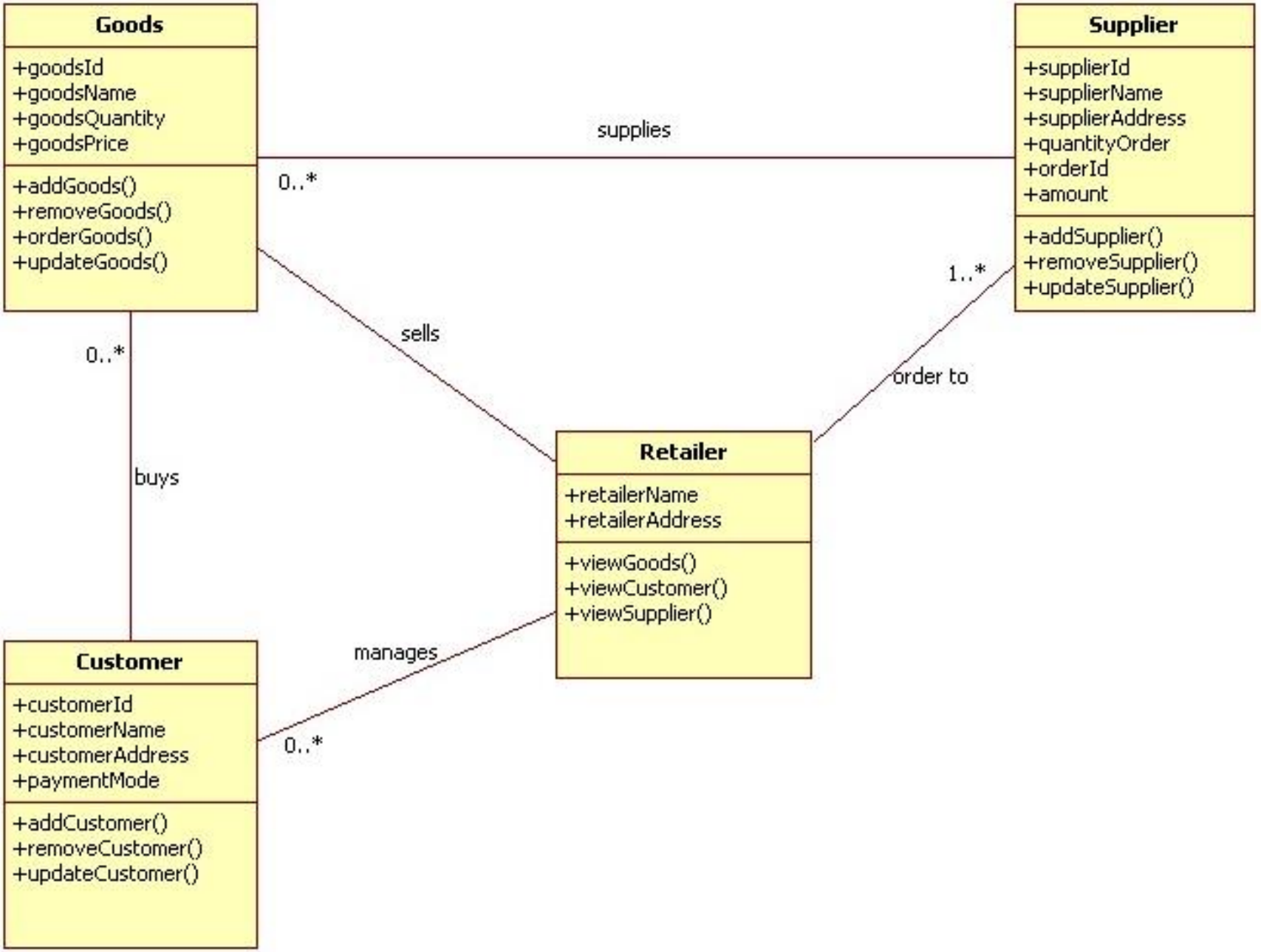
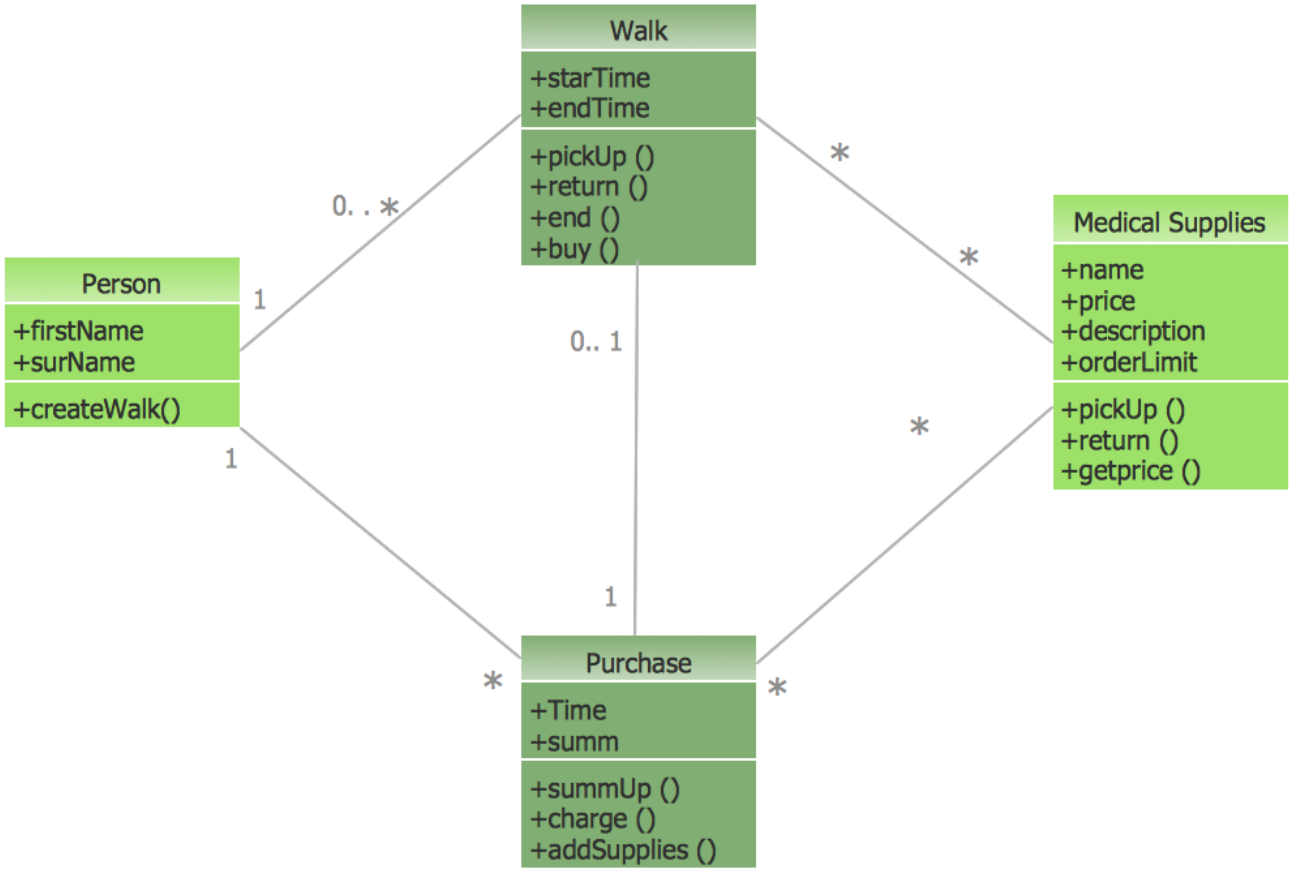
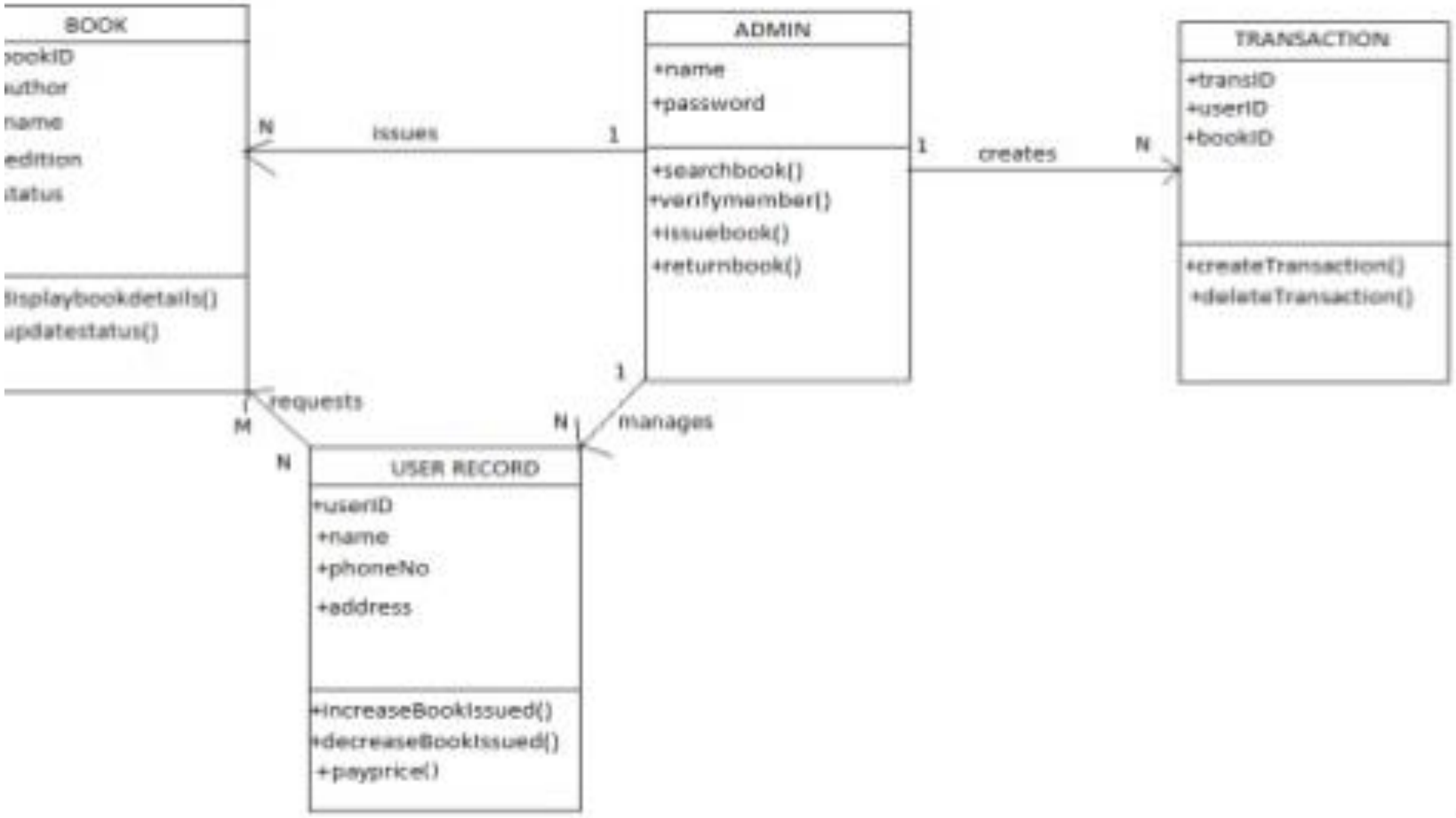


Library system


Goods
+goodsId
+goodsName
+goodsQuantity
+goodsPrice
+addGoods()
+removeGoods()
+orderGoods()
+updateGoods()
supplies
0..*
Supplier
+supplierId
+supplierName
+supplierAddress
+quantityOrder
+orderId
+amount
+addSupplier()
+removeSupplier()
+updateSupplier()
1..*
sells
0..*
order to
buys
Retailer
+retailerName
+retailerAddress
+viewGoods()
+viewCustomer()
+viewSupplier()
manages
Customer
+customerId
+customerName
+customerAddress
+paymentMode
+addCustomer()
+removeCustomer()
+updateCustomer()
0..*


Walk
+starTime
+endTime
+pickUp ()
+return ()
+end ()
+buy ()
Person
+firstName
+surName
+createWalk()
Medical Supplies
+name
+price
+description
+orderLimit
+pickUp ()
+return ()
+getprice ()
Purchase
+Time
+summ
+summUp ()
+charge ()
+addSupplies ()
0. . *
1
1
0.. 1
*
*
*
*
1

SS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



In UML modeling, *interfaces* are model elements that define sets of operations that other model elements, such as classes, or components must implement. An implementing model element realizes an interface by overriding each of the operations that the interface declares.

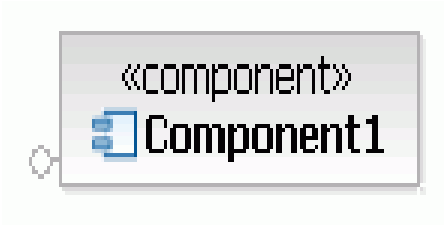
You can use interfaces in class diagrams and component diagrams to specify a contract between the interface and the classifier that realizes the interface. Each interface specifies a well-defined set of operations that have public visibility. The operation signatures tell the implementing classifiers what kind of behavior to invoke, but not how they should invoke that behavior. Many classifiers can implement a single interface, each one providing a unique implementation.

- Interfaces support the hiding of information and protect client code by publicly declaring certain behavior or services. Classes or components that realize the interfaces by implementing this behavior simplify the development of applications because developers who write client code need to know only about the interfaces, not about the details of the implementation. If you replace classes, or components that implement interfaces, in your model, you do not need to redesign your application if the new model elements implement the same interfaces.

- An interface typically has a name that reflects the role that it plays in an application. A common convention is to prefix the name of the interface with a forward slash to indicate that a model element is an interface.
- As the following figures illustrate, the diagram editor displays an interface in the following ways:Class rectangle symbol that contains the keyword «interface». This notation is also called the internal or class view.
- Use the class shape when you need to model the details of the interface. Compartments in the class shape display information about the attributes, operations, and signal receptions of the interface.

- Ball and socket notation, in which the implementation dependency from a classifier to the provided interface is displayed as a circle (ball) and the usage dependency from a classifier to the required interface is displayed as a half-circle (socket). This notation is also called the external view.Provided interface (circle shape)Required interface (socket shape)

Provided interface (circle shape)



Required interface(socket shape)