classes, interfaces and their relationships. When you draw class diagrams, you can use only those class attributes that are important in the current context of your application and display related classes in the same diagrams. It gives the static view of the system. For more details refer to section 2.3.

3) Generally, three types of relationships exist in UML diagrams: Dependencies, Generalization, and Association. **Dependencies** are those relationships which occur between two entities when changes occur in the specification of one element may affect another element. **Generalization** is relationships identified in the class-subclass scenario; it is presented when one entity inherits from another. **Association** is "a-part-of" relationships between classes. It describes relationships between classes in which objects of one class is connected to objects of other class for communicating data. Aggregation is a type of association that specifies a "whole-part" relationship between the aggregate (whole) and the component part.

Check Your Progress 2

1. A class diagram is a static structure diagram in the Unified Modeling Language (UML). It defines the structure of a system by displaying the system's classes, their attributes, operations and the relationships (Dependencies, Generalization, and Association) between objects. You can define some advanced features of the classes like visibility, scope, multiplicity.

   By using visibility feature, a classifier can be accessed by the other classifiers. You can define visibility of classifier's attributes and operations by using plus sign (+) for public (visible to all elements), negative sign (-) for private (only the classifier itself), hash sign (#) for protected (only by the descendants and classifier itself) and tilde (~) for package (all classifier with the same package). The default visibility of a classifier is public in UML.

   Multiplicity describes how many instances of one class may communicate to a single instance of other class. You can display a multiplicity of exactly one (1), zero or one (0..1), many (0..*), or one or more (1..*). The scope feature signifies that an attribute or an operation has their existence in all the instances of the classifier or only one copy is available and is shared across all the instances of the classifier. There are two types of scope specifiers such as "instances" and "class scope". The class scope feature is written as underlined on a class diagram.

2. Generalization is "a-kind-of" relationship. It defines a base or parent class which have some properties functions. A new class will be derived from this base class and it is called child or subclass. The subclass will have access to all the properties and functions of the base or parent class. For example: In a vehicle-car class relationship, vehicle class is parent class while car class is child class.

   Association is a special kind of relationship and it is called a "has-a" relationship. It describes relationships between classes in which objects of one class are connected to other class objects for communicating data. When you connect just two classes is called a binary association. Associations can also have more than two classes known as n-ary associations. Each association can have a name, role, multiplicity and aggregation. The special form of association, called aggregation, specifies a "whole-part" relationship between the aggregate (whole) and component part. For example, a paragraph is part of a word document.

ING

3. A qualified association is a UML concept equivalent to a programming concept called associative arrays, maps, and dictionaries. A qualified association has a qualifier that communicates two object classes with a qualifier key. The qualifier is a special attribute that decreases the multiplicity at the target end of the association. For example, Bank has many customers. They may be distinguished in a bank by their account number. The qualified association is shown as in the following figure.

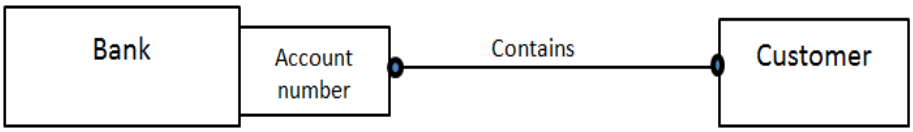

Figure 2.21: Association – Bank and Customer

Check Yo

1. Modeling is a tested and well-known technique that is used to build a model. Model is a blueprint of the actual system that needs to be built. The model supports visualising the system, identifying the system's structural and behaviour, and giving assistance to make templates for constructing the system. This helps in preparing the design document of the system. You can define any number of operations in the interface to provide services to a class or component. You can neither define any attributes nor any implementation of the operations in the interface.
2. It is a kind of structural diagram that displays the system's structure at the level of packages. A package is used for organizing elements into groups. This diagram uses two main types of dependencies: import and access. The dependencies are shown by using dotted arrows like in the figure drawn. In Import dependency, functionality has been imported from one package to another, while the Access dependency designates that one package needs assistance from the functions of another package.
3. An object diagram is a UML structural diagram. It displays the instances of the classifiers in models. Object diagrams display specific instances of those classifiers and the links between those instances at a point in time. The notation of object diagrams is similar to that used in class diagrams. For the figure of object diagram, please refer to section 2.7.2 .

2.10 REFERENES/FURTHER READING

- Grady Booch, James Rumbaugh and Ivar Jacobson,“ The Unified Modeling Language User Guide”, 2nd Edition, Addison-Wesley Object Technology Series, 2005.
- Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston, “Object-Oriented Analysis and Design with Applications,” 3rd Edition, Addison-Wesley, 2007.
- James Rumbaugh, Ivar Jacobson, and Grady Booch, “Unified Modeling Language Reference Manual,” 2nd Edition, Addison-Wesley Professional, 2004.
- John W. Satzinger, Robert B. Jackson, and Stephen D. Burd, “Object-oriented analysis and design with the Unified process,” 1st Edition, Cengage Learning India, 2007.

- Brett McLaughlin, Gary Pollice,and Dave West, “Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to OOA&D,” Shroff Publisher,First edition,2006.
- https://www.uml.org/
- https://www.uml-diagrams.org/index-examples.html
- https://www.uml-diagrams.org/classifier.html
- https://www.uml-diagrams.org/dependency.html
- https://www.uml-diagrams.org/association.html
- https://www.uml-diagrams.org/package-diagrams-overview.html
- https://www.uml-diagrams.org/package-diagrams.html