MODULE 4

1. What is CORBA, and who defined its standard?

- Answer: CORBA is the Common Object Request Broker Architecture, and its standard is defined by the Object Management Group (OMG).

2. What is the Interface Definition Language (IDL) in the context of CORBA?

- Answer: IDL is an implementation-independent language used to describe the interfaces of remote-capable objects in CORBA.

3. What are the standard mappings defined by OMG for IDL interfaces?

- Answer: The OMG has defined standard mappings for converting IDL interfaces into C++ classes, C code, and Java classes, among other things.

4. How does Java IDL compare to RMI in terms of accessing remote objects?

- Answer: Java IDL, like RMI, provides a way to access remote objects over the network. However, unlike RMI, CORBA objects can be accessed by clients implemented in any language with an IDL binding.

5. What are some of the distributed object services specified by the CORBA standard?

- Answer: The CORBA standard includes specifications for distributed object services such as an object naming service, a security policy service, and persistent object services.

6. What is the Object Management Group (OMG), and what is its purpose?

- Answer: OMG is a non-profit consortium formed in 1989 to promote the theory and practice of object technology for distributed operating systems. It provides a common architectural framework for object-oriented applications based on widely available interface specifications.

7. How does the CORBA architecture for distributed objects compare to Java RMI?

- Answer: Both CORBA and Java RMI involve generating client stubs and server skeletons to invoke methods on remote objects. However, CORBA is more extensive in its specification and language-independent.

8. What is the role of the Interface Definition Language (IDL) in CORBA?

- Answer: IDL is used to describe data types and interfaces in CORBA, and it is independent of any specific programming language.

9. What is the Object Request Broker (ORB) in the CORBA architecture?

- Answer: The ORB is responsible for handling interactions between remote objects and applications. It facilitates requests between client and server objects.

10. What is the purpose of the Naming Service in CORBA?

- Answer: The Naming Service provides a directory naming structure for remote objects, allowing them to be found by clients on the network.

11. What is the role of the Object Adaptor in the CORBA architecture?

- Answer: The Object Adaptor connects a server object to the CORBA runtime environment, allowing server objects to interact with the ORB and vice versa.

12. What is the Internet Inter-ORB Protocol (IIOP), and what is its significance?

- Answer: IIOP is an inter-ORB protocol based on TCP/IP and is the most commonly used CORBA communication protocol. It allows different ORBs to communicate over the Internet.

13. How does CORBA support different programming languages with respect to remote object communication?

- Answer: CORBA is language-independent and provides a way to specify and communicate with remote objects in languages that might not have built-in support for objects. It includes details about Dynamic Invocation and Dynamic Skeleton Interfaces to support such languages.

14. How is the Naming Service directory structured, and how are objects represented within it?

- Answer: The directory structure starts with a root node, with subnodes and leaf objects defined within it. Objects are stored by name at the leaves of the tree, and their fully qualified names include all parent nodes.

15. What is the primary goal of the Object Management Group (OMG) in the context of CORBA?

- Answer: The OMG aims to provide a common architectural framework for object-oriented applications based on widely available interface specifications, promoting the use of object technology in distributed operating systems.

Q1: What is CORBA and what does it stand for?

A1: CORBA stands for Common Object Request Broker Architecture. It is a standard defined by the Object Management Group (OMG) that describes an architecture, interfaces, and protocols for distributed objects to interact with each other.

Q2: What is the purpose of the Interface Definition Language (IDL) in CORBA?

A2: IDL is an implementation-independent language for describing the interfaces of remote-capable objects in CORBA. It provides a way to define data types and interfaces that can be used by distributed objects regardless of the programming language they are implemented in.

Q3: How does Java IDL facilitate communication between Java programs and distributed objects in CORBA?

A3: Java IDL provides an implementation of the standard IDL-to-Java mapping. It allows Java programs to communicate with distributed objects in CORBA. Java IDL generates classes that use the underlying CORBA framework to communicate with remote clients, making it easier to implement and export distributed objects in Java.

Q4: What is the role of the Object Request Broker (ORB) in the CORBA architecture?

A4: The ORB is the core component of the CORBA architecture. It handles all interactions between remote objects and the applications that use them. ORBs are responsible for making requests happen between client and server objects. Client ORBs provide a stub for a remote object, and requests made on the stub are transferred to the ORB servicing the implementation of the target object.

Q5: How does the Naming Service work in CORBA?

A5: The Naming Service provides a directory naming structure for remote objects in CORBA. It organizes objects in a tree structure with branches called naming contexts. Each leaf object has bindings to specific names. Clients can find objects by giving their names relative to the appropriate naming context. The Naming Service allows remote clients to locate and access objects on the network.

Q6: What is the purpose of the Object Adaptor in CORBA?

A6: The Object Adaptor plugs a server object into a specific CORBA runtime environment. It facilitates communication between the server object and the core functionality of the ORB. There are different versions of Object Adaptors, including Basic Object Adaptor (BOA) and Portable Object Adaptor (POA), which support server objects in various CORBA implementations.

Q7: What is the significance of the Interface Definition Language (IDL) being language-independent in CORBA?

A7: IDL being language-independent means that it can describe interfaces and data types in a way that is not tied to a specific programming language. This allows objects implemented in different programming languages to interact seamlessly in a distributed CORBA environment. Standard mappings or bindings from IDL to specific programming languages ensure interoperability between objects implemented in different languages.

Q8: What is the Internet Inter-ORB Protocol (IIOP) in the context of CORBA?

A8: IIOP is an inter-ORB protocol based on TCP/IP. It allows communication between ORBs running on different machines over the Internet. IIOP enables messages, such as method requests and return types, to be transmitted between cooperating ORBs. It ensures that ORBs implemented in different languages can communicate as long as they adhere to the CORBA standard and use a standard communication protocol like IIOP.

2/3 marks questions

Essay Questions and Answers on CORBA:

Question 1:

What is the Common Object Request Broker Architecture (CORBA), and how does it relate to the Java IDL API?

Answer 1:

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that provides an architecture, interfaces, and protocols for distributed objects to interact with each other. It includes the Interface Definition Language (IDL) for describing the interfaces of remote-capable objects. The Java IDL API, introduced in Version 1.2 of the Java 2 platform, serves as an interface between Java programs and distributed objects and services built using CORBA. It allows Java programs to access remote objects and services, making them accessible to other CORBA clients.

Question 2:

How does Java IDL differ from RMI, and what are the advantages of using Java IDL for distributed object communication?

Answer 2:

Java IDL and RMI (Remote Method Invocation) both provide a way to access remote objects over the network, but there are differences between them. Unlike RMI, Java IDL is not limited to Java clients and can be accessed by clients implemented in any language with an IDL binding (C, C++, Ada, etc.). This language independence is one of the advantages of using Java IDL. Additionally, Java IDL allows you to create and publish objects through a naming/directory service, making them discoverable by remote clients. RMI, on the other hand, primarily supports Java clients. The choice between RMI and Java IDL depends on the requirements of the project and the need for language interoperability.

Question 3:

What are the major components of the CORBA architecture, and how do they work together to facilitate distributed object communication?

Answer 3:

The major components of the CORBA architecture include:

- Interface Definition Language (IDL): It defines the data types in CORBA and provides language-independent descriptions of interfaces.

- Object Request Broker (ORB): It handles interactions between remote objects and applications, serving as a middleware to route method requests between clients and servers.

- Object Adaptor: It connects server objects to the CORBA runtime environment, allowing the ORB to pass client requests and lifecycle notifications to the server objects.

- Naming Service: It provides a directory naming structure for remote objects, allowing clients to discover and access objects on the network.

- Inter-ORB Communication: It includes protocols like IIOP, which enable communication between different ORBs on the network, ensuring that messages, method requests, and return types can be transmitted effectively.

These components work together to enable the creation, discovery, and communication of distributed objects within the CORBA architecture.

Question 4:

Why does CORBA include a detailed specification of an object model, and how does it support languages with different built-in features?

Answer 4:

CORBA includes a detailed specification of an object model because it was designed to be a language-independent distributed object standard. Unlike RMI, CORBA aims to support languages with varying built-in features, some of which may not directly support objects. The extra details in the CORBA specification are necessary to ensure that non-object-oriented languages can take advantage of CORBA. Additionally, CORBA specifies features like the Dynamic Invocation Interface and Dynamic Skeleton Interface to facilitate operations in languages that lack their own facilities for these tasks. In languages with built-in support for object interfaces, such as Java, a mapping is defined to bridge the gap between the language's features and the CORBA specification, ensuring interoperability.

These specifications and mappings allow CORBA to be a versatile and language-independent standard for distributed object communication.

ESSAY QUESTION ABOUT CORBA ARCHITECTURE

The paragraph provides information about the architecture of CORBA (Common Object Request Broker Architecture). Here's the answer to your 5-mark question:

The CORBA architecture consists of several key components and services that facilitate the communication and interaction of distributed objects:

1. Interface Definition Language (IDL): IDL is an implementation-independent language used to describe the interfaces of remote-capable objects. It serves as the primary means of defining data types in CORBA and is independent of any particular programming language. Mappings from IDL to specific programming languages are defined and standardized as part of the CORBA specification.

2. Object Request Broker (ORB): The ORB is the core component of the CORBA architecture. It is responsible for handling all interactions between remote objects and the applications that use them. Each machine involved in a CORBA application must have an ORB running to facilitate communication between processes on that machine and remote processes. The client ORB provides a stub for a remote object, and requests made on the stub are transferred to the ORB servicing the implementation of the target object. The ORB manages the communication infrastructure.

3. The Object Adaptor: The Object Adaptor provides a general facility for integrating server objects into a specific CORBA runtime environment. It enables server objects to interact with the core functionality of the ORB and facilitates the passing of client requests and lifecycle notifications to the server object. Depending on the CORBA version and JDK, there can be different Object Adaptor interfaces (e.g., BOA or POA).

4. Naming Service: The CORBA Naming Service provides a directory naming structure for remote objects, allowing them to be located on the network. Objects are stored by name at the leaves of the tree, and the tree structure starts with a root node. Each branch in the directory tree is called a naming context, and leaf objects have bindings to specific names. The Naming Service is essential for locating and accessing remote objects in a CORBA environment.

5. Inter-ORB Communication: CORBA includes specifications for inter-ORB communication protocols that enable the transmission of object requests between various ORBs running on the network. These protocols are independent of the specific ORB implementations and handle differences in machine-level byte ordering and alignment. The Internet Inter-ORB Protocol (IIOP) is a widely used inter-ORB protocol based on TCP/IP.

These components together form the core of the CORBA architecture, allowing for language-independent distributed object interactions, providing a means to define object interfaces, and supporting naming and communication between distributed objects.

This architecture enables the development of distributed applications in different programming languages, making CORBA a versatile and extensive standard for distributed computing.