Nikki Wagholikar(2019240071) Batch: A
Saish Patil(2019240069) Class: TEIT

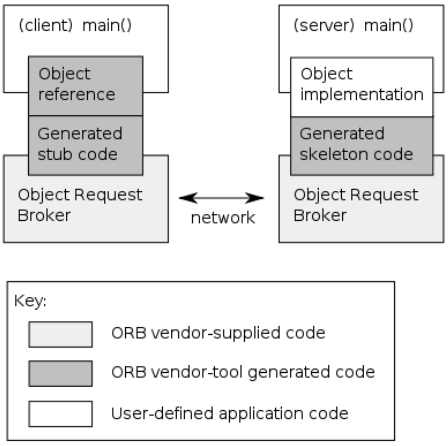
Experiment 6

Aim: To develop a distributed application using CORBA.

Theory:

- CORBA(**Common Object Request Broker Architecture**) enables communication between software written in different languages and running on different computers.
- CORBA normalizes the method-call semantics between application objects residing either in the same address-space (application) or in remote address-spaces (same host, or remote host on a network.
- CORBA uses an interface definition language (IDL) to specify the interfaces that objects present to the outer world. CORBA then specifies a *mapping* from IDL to a specific implementation language like C++ or Java
- The CORBA specification dictates there shall be an ORB through which an application would interact with other objects.
- This is how it is implemented in practice:

1. The application simply initializes the ORB, and accesses an internal *Object Adapter*, which maintains things like reference counting, object (and reference) instantiation policies, and object lifetime policies.
2. The Object Adapter is used to register instances of the *generated code classes*. Generated code classes are the result of compiling the user IDL code, which translates the high-level interface definition into an OS- and language-specific class base for use by the user application. This step is necessary in order to enforce CORBA semantics and provide a clean user process for interfacing with the CORBA infrastructure.



- In addition to providing users with a language and a platform-neutral remote procedure call (RPC) specification, CORBA defines commonly needed services such as transactions and security, events, time, and other domain-specific interface models.

Case Study: A Gift store where you can buy gifts from the catalog provided and it will provide you with the amount and bill at the end.

Code:

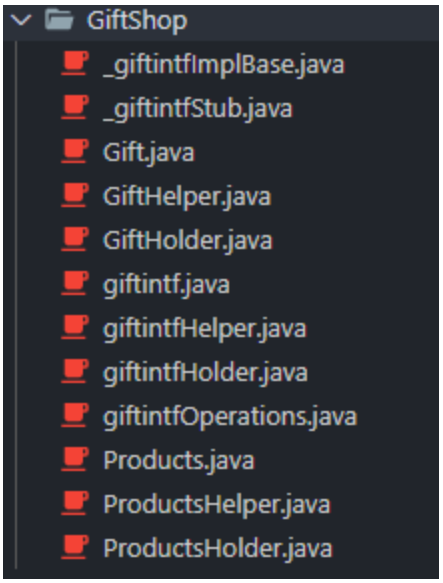
1. Create shop.idl file

```
module GiftShop{
  struct Gift{
    string name;
    long count;
    long price;
  };

  struct Products{
    Gift gifts[10];
  };

  interface giftintf{
      string check_gift(in string gift_id);
      string process_gift(in string gift_id, in long quantity);
      long total_price();
      string pay_bill();
      };
};
```

2. Compile the idl using the command:

```
idlj -fAll -oldImplBase shop.idl
```

3. The above command generates the following files:



4. Create class Server.java

```
import GiftShop.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

class Server extends _giftintfImplBase {

  static Map<String, Integer> gift_list;
  static int[] prices;
  static int total;
  static int store;

  public Server() {
    total = 0;
    gift_list = new LinkedHashMap<String, Integer>();
    prices = new int[10];
    gift_list.put("Polaroids", 100);
    gift_list.put("Keychains", 20);
    gift_list.put("Earrings", 20);
    gift_list.put("PhotoFrame", 20);
    gift_list.put("GiftBox", 20);
    gift_list.put("Cards", 20);
```

```
    gift_list.put("Painting", 20);
    gift_list.put("ScrapBook", 20);
    gift_list.put("Stickers", 100);
    gift_list.put("Bookmarks", 20);

    prices[0] = 5;
    prices[1] = 50;
    prices[2] = 100;
    prices[3] = 120;
    prices[4] = 200;
    prices[5] = 100;
    prices[6] = 50;
    prices[7] = 150;
    prices[8] = 20;
    prices[9] = 25;
  }

  // checking for availabilty for gifts
  public String check_gift(String gift_id) {
    store = -1;
    int i = 0;
    if (!gift_list.containsKey(gift_id)) {
      return "Item not present";
    }

    String s = "";
    for (String ik : gift_list.keySet()) {
      if (ik.equals(gift_id)) {
        s =
          "Item present with total available quantity: " +
          gift_list.get(gift_id) +
          "\nPrice:" +
          prices[i];
        store = i;
        break;
      }
      i++;
    }
    return s;
  }
```

```
// add to cart
public String process_gift(String gift_id, int quantity) {
  int q = gift_list.get(gift_id);
  String s = "";
  if (q >= quantity) {
    total += prices[store] * quantity;
    gift_list.put(gift_id, gift_list.get(gift_id) - quantity);
    System.out.println(
      "Item " + gift_id + " ordered with quantities " + quantity
    );
    s = "Item added to cart";
  }
  return s;
}

//return total prices
public int total_price() {
  return total;
}

public String pay_bill() {
  System.out.println("Payment successful with amount " + total);
  total = 0;
  return "Payment successful!\n";
}

public static void main(String[] args) {
  try {
    ORB orb = ORB.init(args, null);
    Server lbRef = new Server();
    orb.connect(lbRef);
    org.omg.CORBA.Object objRef = orb.resolve_initial_references(
      "NameService"
    );
    NamingContext ncRef = NamingContextHelper.narrow(objRef);
    NameComponent nc = new NameComponent("gifts", "");
    NameComponent path[] = { nc };
    ncRef.rebind(path, lbRef);
    System.out.println("Server started!");
```

```
      Thread.currentThread().join();
    } catch (Exception e) {
      System.err.println(e);
    }
  }
}
```

5. Create Client.java file.

```
import GiftShop.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.util.*;
class Client{
  public static void main(String[] args){
    try{
      int i;
      String j;
      Scanner sc=new Scanner(System.in);
      String s,b,a;
      ORB orb=ORB.init(args,null);
      org.omg.CORBA.Object objRef =
      orb.resolve_initial_references("NameService");
      NamingContext ncRef = NamingContextHelper.narrow(objRef);
      NameComponent nc=new NameComponent( "gifts" , "" );
      NameComponent path[] = {nc} ;
      giftintf lbref=giftintfHelper.narrow( ncRef.resolve(path)
);
      // medintf lbref=medintfHelper.narrow(
ncRef.resolve(path) );

      System.out.println("~~~~~~ Welcome to Hobbies&Love
~~~~~~");
      System.out.println("Gift catalog:");
      System.out.println("1 Polaroids");
      System.out.println("2 Keychains");
      System.out.println("3 Earrings");
```

```
      System.out.println("4 PhotoFrame");
      System.out.println("5 GiftBox");
      System.out.println("6 Cards");
      System.out.println("7 Painting");
      System.out.println("8 ScrapBook");
      System.out.println("9 Stickers");
      System.out.println("10 Bookmarks");


System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

      do{
        int p=0;
        System.out.println( ++p +".Check gift items");
        if(lbref.total_price() > 0){
              System.out.println( ++p +".Proceed to Payment");
        }
        System.out.println("0.Exit");
        System.out.print("Enter choice: ");
        i=sc.nextInt();
        switch(i){
          case 1:System.out.print("Enter gift item name: ");
          j=sc.next();
          s=lbref.check_gift(j);
          System.out.println(s);
          System.out.println("If you want to  confirm type
'yes' else 'no'");
          b=sc.next();    a=b.toLowerCase();
          if(a.equals("yes")){
             System.out.print("Enter number of quantities: ");
             int y=sc.nextInt();
             s=lbref.process_gift(j,y);
             if(s.equals("")){
               s="Sorry this gift item is not in stock!";
             }
             System.out.println(s);
          }
          break;

          case 2:int pay=lbref.total_price();
```

```
          System.out.println("Total amount to be paid: "+pay);
          System.out.println("If you want to pay type 'yes'
else 'no' ");
            b=sc.next();    a=b.toLowerCase();
          if(a.equals("yes")){
             s=lbref.pay_bill();
             System.out.println(s);
          }
          break;

          default:
          if(lbref.total_price() > 0){
            System.out.println("Your cart amount:
"+lbref.total_price());
           System.out.print("Do you want to pay? ");
           b=sc.next();   a=b.toLowerCase();
           if(a.equals("yes")){
                 s=lbref.pay_bill();
                 System.out.println(s);
           }
          }

          break;
        }
      }while(i!=0);
    }catch(Exception e){
      e.printStackTrace();
    }
  }
}
```

6. Compile all files using the command:

javac *.java

7. Start ordb from terminal using command:

orbd -ORBInitialPort 1050

8. Start Server using the command:

java Server -ORBInitialPort 1050 -ORBInitialHost localhost

10. Start Client using the command:

java Client -ORBInitialPort 1050 -ORBInitialHost localhost

Output:

- **ordb**

```
PS E:\TE IT\SEM 6\DS Distributed Systems\Lab\Exp 6\GiftShop> orbd -ORBInitialPort 1050
```

- **Server**

```
PS E:\TE IT\SEM 6\DS Distributed Systems\Lab\Exp 6\GiftShop> java Server -ORBInitialPort 1050 -ORBInitialHost localhost
Server started!
Item Polaroids ordered with quantities 10
Payment successful with amount 50
```

- **Client**

```
PS E:\TE IT\SEM 6\DS Distributed Systems\Lab\Exp 6\GiftShop> java Client -ORBInitialPort 1050 -ORBInitialHost localhost
~~~~~~~ Welcome to Hobbies&Love ~~~~~~~
Gift catalog:
1 Polaroids
2 Keychains
3 Earrings
4 PhotoFrame
5 GiftBox
6 Cards
7 Painting
8 ScrapBook
9 Stickers
10 Bookmarks
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1.Check gift items
0.Exit
Enter choice: 1
Enter gift item name: Polaroids
Item present with total available quantity: 100
Price:5
If you want to  confirm type 'yes' else 'no'
yes
Enter number of quantities: 10
Item added to cart
1.Check gift items
2.Proceed to Payment
0.Exit
Enter choice: 2
Total amount to be paid: 50
If you want to pay type 'yes' else 'no'
yes
Payment successful!

1.Check gift items
0.Exit
Enter choice: 0
PS E:\TE IT\SEM 6\DS Distributed Systems\Lab\Exp 6\GiftShop>
```

Conclusion: Thus we have developed a component for shopping gift articles using CORBA.