Task 8: Service implementation

1.Create CourierUserServiceImpl class which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompany. This variable can be used to access the Object Arrays to access data relevant in method implementations.

CourierUserServiceImpl class

In this class the implementation of the methods is achieved using arrays and holds a variable of type CourierCompany.

```
package DAO;

import Entities.Courier;
import Entities.CourierCompany;
import Interface.ICourierUserService;

import java.util.Map;

public class CourierUserServiceImpl implements
ICourierUserService {

   CourierCompany companyObj=new CourierCompany();


   public CourierUserServiceImpl(CourierCompany companyObj) {
      this.companyObj = companyObj;
   }
   //This class mainly implements all the methods based on arrays

   @Override
   public String placeOrder(Courier courier) {
      // Add courier to array in companyObj
      Courier[] couriers = companyObj.getCouriers();
      int count = companyObj.getCourierCount();
```

```
        if (count < couriers.length) {
            couriers[count] = courier;
            companyObj.setCourierCount(count + 1);
            return courier.getTrackingNumber();
        } else {
            return "Courier list is full.";
        }
    }
    @Override
    public String getOrderStatus(Courier courier) {
        return courier.getStatus();
    }

    @Override
    public boolean cancelOrder(Courier courier, String
trackingNumber) {
        if (courier != null &&
courier.getTrackingNumber().equals(trackingNumber)) {
                return true;
            }
            return false;
    }

    @Override
    public Courier[] getAssignedOrder(Courier courierobj, int empid) {
        Courier[] assigned = new Courier[1];
        assigned[0] = courierobj;
        return assigned;
    }
    //This Method is not used here because it's implementation is based
on map

    @Override
    public Map<Integer, Courier> getAssignedOrderMap(Courier
courierobj, int empid) {
```

```
        return Map.of();
    }}
```

2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and implements ICourierAdminService interface.

CourierAdminServiceImpl class

Here the CourierAdminServiceImpl class achieves the logic by arrays and it is inherited from CourierUserServiceImpl(Base Class) and implements ICourierAdminService interface.

```
package DAO;

import Entities.Employee;
import Entities.CourierCompany;
import Interface.ICourierAdminService;

import java.util.Random;

public class CourierAdminServiceImpl extends
CourierUserServiceImpl implements ICourierAdminService {

    public CourierAdminServiceImpl(CourierCompany companyObj)
{
        super(companyObj);
    }

    @Override
    public int addCourierStaff(Employee emp) {
        Random random=new Random();
        int empid=random.nextInt(41)+120;
        //This method logic is same as methodin CourierAdminService
class,
        // But it is specifically used for Array based implementation
```

```
        Employee[] newaddedemp=new Employee[15];
        int count=0;
        newaddedemp[count++]=emp;
        return empid;
    }
}
```

3. Create CourierAdminServiceCollectionImpl class which inherits from CourierUserServiceColectionImpl and implements ICourierAdminService interface.

CourierAdminServiceCollectionImpl class

This implementation class mainly focuses on collections and achieve logic using collections. Also this class inherits from CourierUserServiceColectionImpl and implements ICourierAdminService interface.

```
package DAO;

import Entities.CourierCompanyCollection;
import Entities.Employee;
import Interface.ICourierAdminService;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public  class CourierAdminServiceCollectionImpl extends
CourierUserServiceCollectionImpl implements
ICourierAdminService {

    public
CourierAdminServiceCollectionImpl(CourierCompanyCollection
companyObj) {
        super(companyObj);
    }
```

```
    @Override
    public int addCourierStaff(Employee emp) {
        Random random=new Random();
        int empid=random.nextInt(41)+120;
        //This method logic is same as method in CourierAdminService
class,
        // But it is specifically used for Collection based implementation
        List<Employee> newaddedemp=new ArrayList<>();
        newaddedemp.add(emp);
        return empid;
    }
}
```

These implementation classes are used inorder to achieve flexibility,like when the user wants to perform static updates then operation can be done using arrays,similarly for dynamic updates operation can be done using collections.