Question 1 [40 points] {55 minutes}

Suppose you need to make a relatively simple CRUD (Create, Read, Update, Delete) application that manages employees. The application has the following requirements:

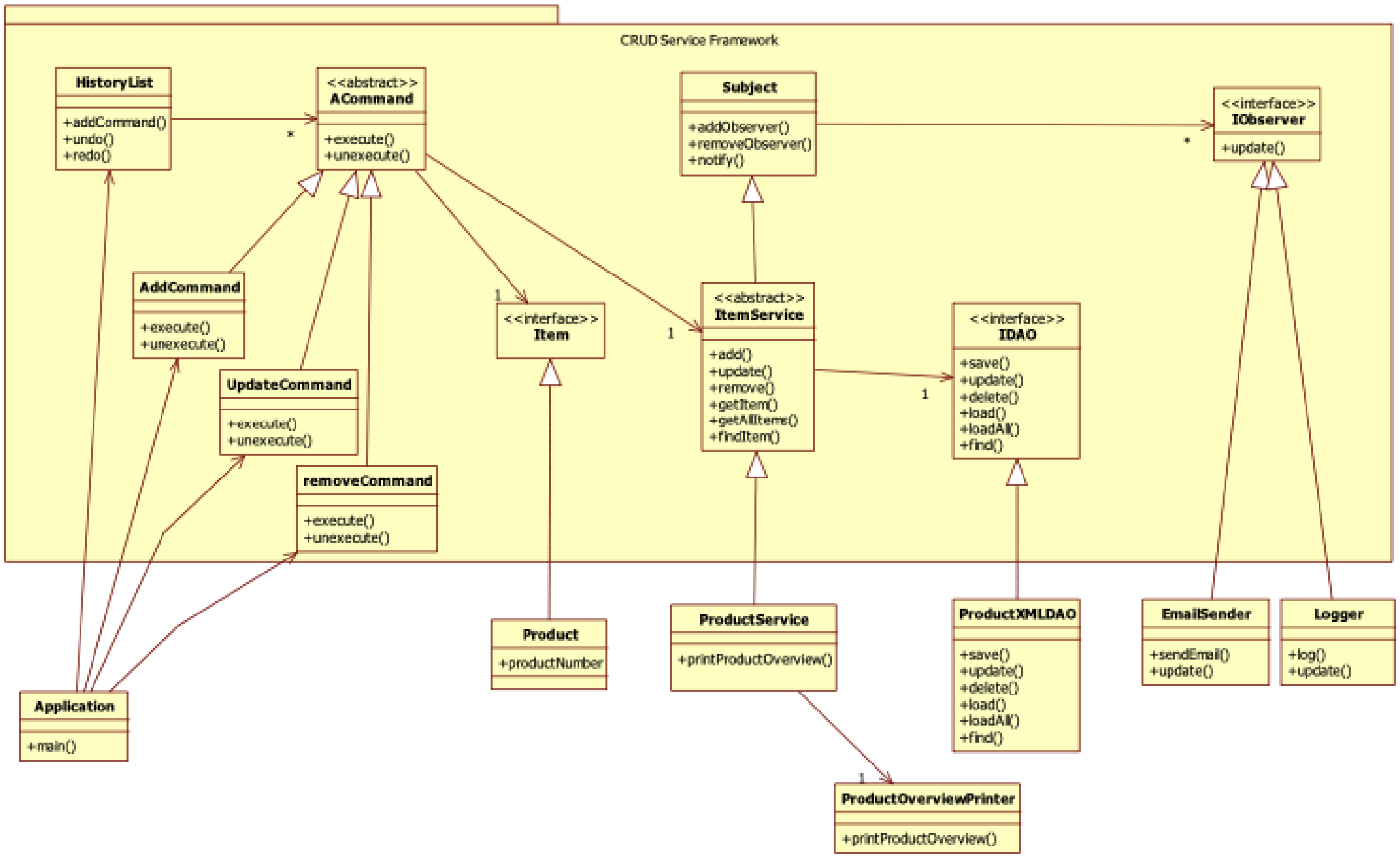
- You can add new employees, update existing employees, remove employees, view an employee, search employees and get a list of all employees.
- You should be able to undo/redo the add, update and delete action
- All employees are stored in the database
- You want to log all add, update and delete actions in a logfile

Just when you want to start with the design of this application you get a request from another customer to write a CRUD ProductService application that manages products.

This application has the following requirements:

- You can add, update, remove and view a product.
- You can search products.
- You should be able to undo/redo the add, update and delete action
- All products are stored in an XML file
- You need to send an email to the sales manager whenever a product is added or deleted.
- You want to log all CRUD actions in a logfile
- You should be able to print an overview of all available products.

Because you need to implement 2 similar CRUD applications you decide to design a general CRUDService framework so that you can reuse this framework for both CRUD applications. This CRUDService framework should be very flexible so that we can reuse it for other CRUD-like applications with similar requirements.


CRUD Service Framework
HistoryList
+addCommand()
+undo()
+redo()
*
<<abstract>>
ACommand
+execute()
+unexecute()
Subject
+addObserver()
+removeObserver()
+notify()
*
<<interface>>
IObserver
+update()
AddCommand
+execute()
+unexecute()
UpdateCommand
+execute()
+unexecute()
removeCommand
+execute()
+unexecute()
1
<<interface>>
Item
1
<<abstract>>
ItemService
+add()
+update()
+remove()
+getItem()
+getAllItems()
+findItem()
1
<<interface>>
IDAO
+save()
+update()
+delete()
+load()
+loadAll()
+find()
Product
+productNumber
ProductService
+printProductOverview()
ProductXMLDAO
+save()
+update()
+delete()
+load()
+loadAll()
+find()
EmailSender
+sendEmail()
+update()
Logger
+log()
+update()
Application
+main()
1
ProductOverviewPrinter
+printProductOverview()

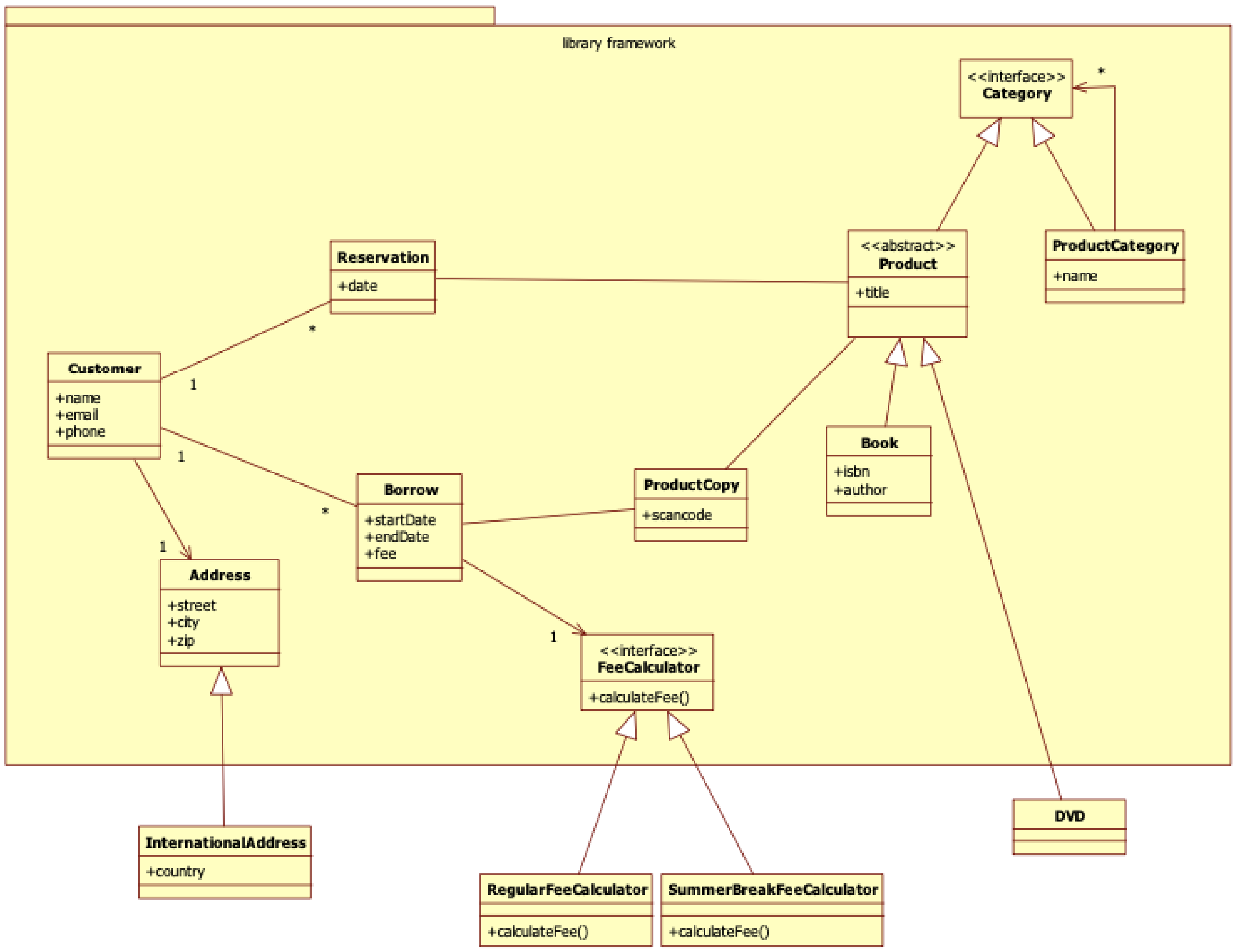
Question 2 [40 points] {45 minutes}

Suppose we need to design a library system framework which allows us to manage Books, borrowings and reservations. Consider the following requirements for this framework:

- The framework should record customer information: name, phone, email, street, city, zip.
- The framework should record book information: title, isbn, author.
- When a customer checks out We should be able to record (checkout-date, return-date, fee) and handle the different books that a customer borrows.
- We should be able to record and handle the reservations of a customer.
- It should be easy to plugin different algorithms to compute the fee for books that are returned to late.
- We can have multiple copies of the same book title. Every copy has a unique scancode.
- The framework should support the ability to create different book categories and subcategories. For example, we can have a category Computer book with subcategory UML books

Draw the class diagram of this library framework. In the same class diagram, show how this library framework is used by a library application with the following requirements:

- The library application should support both Books and DVD's.
- The library application should support 2 different ways to compute the fee for books that are returned too late:
- During the summer break we calculate the fee in a different way as during the regular days.
- The library application should support foreign customers. The application should record the country for these foreign customers.


library framework
<<interface>>
Category
*
<<abstract>>
Product
+title
ProductCategory
+name
Reservation
+date
*
Customer
+name
+email
+phone
1
1
Borrow
+startDate
+endDate
+fee
*
ProductCopy
+scancode
Book
+isbn
+author
1
Address
+street
+city
+zip
1
<<interface>>
FeeCalculator
+calculateFee()
DVD
InternationalAddress
+country
RegularFeeCalculator
+calculateFee()
SummerBreakFeeCalculator
+calculateFee()

Question 3 [15 points] {10 minutes}

Consider the following application that uses a dynamic proxy:

```
public interface IVehicle {
       void start();
}


public class Car implements IVehicle {
      private String name ="Herbie";

      public void start() {
            System.out.println("Car " + name + " started");
      }

}


public class Logger implements InvocationHandler {
      private Object v;

      public Logger(Object v) {
            this.v = v;
      }

      public Object invoke(Object proxy, Method m, Object[] args)
                  throws Throwable {
            System.out.println("Logger: " + m.getName());
            Object object= m.invoke(v, args);
            return object;
      }
}


public class Notifier implements InvocationHandler {
      private Object v;

      public Notifier(Object v) {
            this.v = v;
      }

      public Object invoke(Object proxy, Method m, Object[] args)
                  throws Throwable {
            System.out.println("Notifier: " + m.getName());
```

```
Notifier: start
Logger: start
Car Herbie started
Notifier: start
```