SOLID Report

1. S (Single Responsibility Principle)

Related modules	Description	Improvement
Controller (and methods contains in that package)	Controller is the backbone of program, they have to connect to database, view screen and contains it own methods. If they has to solve much of tasks like their methods, it make 1 class has more responsibilities, violates S principle	We only focus on the main function of controller to call and controls others, to connect database we transfer task for entityDB, and the view screen has its own package for solving, the controller just call these above function.
entityBD	Contains only method for connecting with database	Good design

2. O (Open Closed Principle)

Related modules	Description	Improvement
Entity.media, entity.book, entity.cd, entity.dvd	The entity media can be books, cd, or dvd depends on products that shop sales. We may face with plenty of products that can be add to media each day.	Make the media class can contains more than only 1 products, then we can add more products as we want also remains the media for summaries what user choose,

		not define again each time.

3. L (The Liskov Substitution Principle)

Related modules	Description	Improvement
Entity related to media, shipping, invoice, ...	These class can call each other, if the use call function, they have to use objects of derived classes without knowing it or directly modifies their value	We have to setup the the attributes of each class to private and only provides the get or set method if necessary not public once, then if want to use object, we have to clearly define that object then modify.

4. I (The Interface Segregation Principle)

Related modules	Description	Improvement
The view package (and related class and .fxml file for creating interface for program)	They also has their own relationship and they connect to each other in term of function and meaning of program	Set up the connection and supported function which the controlling belongs to the controller class
The Interbank Interface	Since we may have more than 1 method for payment (not on;y credit card) or some more transaction not only payment like deposit	In the basic meaning, all above class has the same tasks is dealing with possible transaction. Then we can not separated these method into

	and return deposit, each time we want to pay call this method and rewrite each case is time consuming	different interface but also can extends the interface for saving coding resoures

5. D (Dependency Inversion Principle)

Related modules	Description	Improvement
AimsException	AimsException is the main function for dealing with errors of programs. Thar means this function contains all cases of exception, if we defines only 1 fixed comment, user can not know how they got this problem. Moreover, we have to define this methods many times in others parent class, that violates D principle	We define the main abstract class AimsException, then in each case of error, we can extend then override this method. Other methods wants to call to error handling just directly call Exception, we don’t need to rewrite class anymore.
PaymentException	Same with AimsException, PaymentException for dealing with error when transaction occurs. That means we could face with many cases can be	We build abstract class PaymentException then other error handling related will extends this method, this makes no rewrite and parent class will

	happened.	not directly depends on children class.