Payment cohesion review:

Until now, functionality Payment in our project is implemented in the following packages(modules):

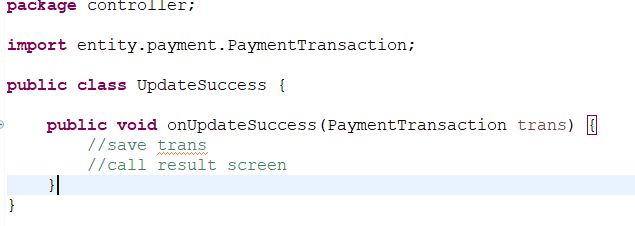
* subsystem package
* entity.payment package
* controller package

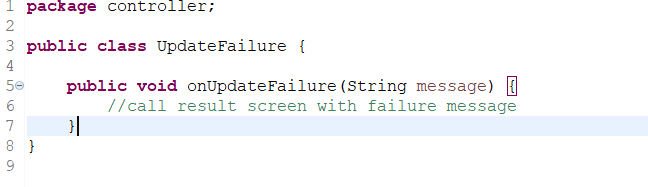
To examine cohesion level of functionality Payment, we has to examine cohesion level of each package as well as its component,

which means we review the degree all the methods and attributes in these components and class in these packages direct towards the responsibility of the modules it belongs to.

1.controller package - PayOrderController class

* In PayOrderController class, the main responsibility of this class is handling pay order action and only payOrder() method supports this. Two remaining classes are not related to the responsibility, so this class is **coincidental cohesion**. To avoid this, we should move onUpdateSuccess() method and onUpdateFailure() method to different classes.





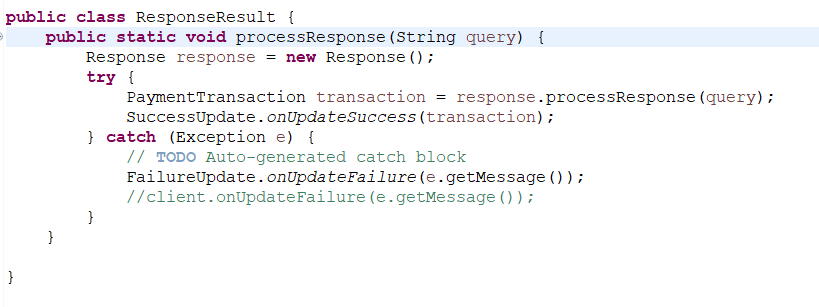
2.entity.payment package

* This package has 1 class which is PaymentTransaction class. This class contain different attributes of information about 1 transaction, a constructor to create object PaymentTransaction and a override method called toString(). toString() is created to provide a way to get information in string format about the transaction. The responsibility of toString() is closely connected to PaymentTransaction so this is **functional cohesion** and should not be modified.

3.subsystem package

* VNPayController class: this class contain 2 methods with different purpose: payOrder() process the information about transaction and create request to VNPay, while processResponse() takes query and handle the result of that query. These two methods only has relationship in logical (both provide a way to service for pay order activity) so this is **logical cohesion**. We can modify the design by taking processResponse to a new class called ResponseResult.

Code modification: new class, static processResponse() method:



* Request class: this class contains constructor and method createPayUrl(). Method createPayUrl() is added to the class since it gets access to attributes of request: amount and content, so this design is **communicational cohesion**. Although this level of cohesion is acceptable, we may optimize the design for future extension: we can create a static class called CreatePayUrl, then create static method createPayUrl(Request req) implementing the same responsibility with the above createPayUrl() method. When the system needs to create a pay url, we may call this method and pass a Request object to it. This design makes the work of maintain createPayUrl method easier and in the future if there is any new kind of request we may implement and run it without duplicate the code of createPayUrl() from old Request class.

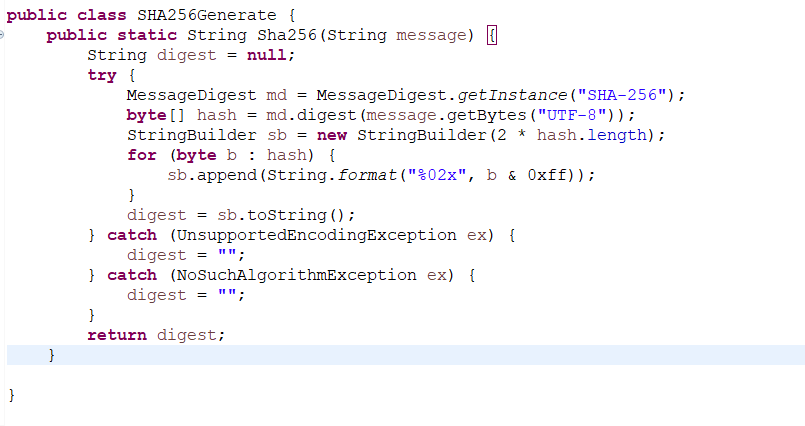
Code modification: new CreatePayUrl class, createPayUrl() method has a Request type argument:



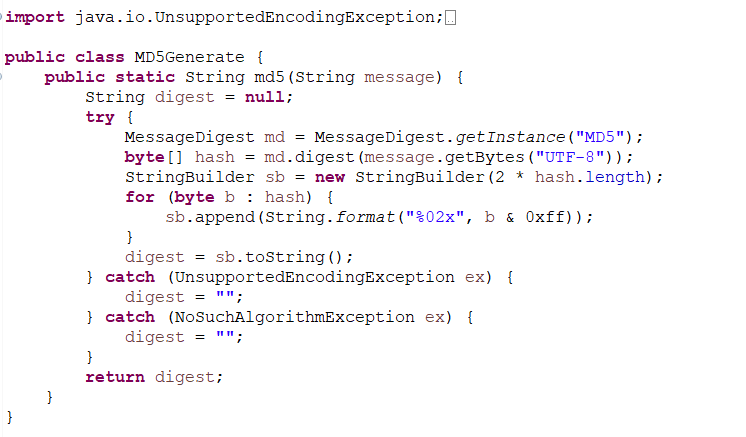
* Response class: the main responsibility of this class is handling query from controller. This class use processResponse() method as the main behaviour for other class to call and supports this method by 2 other classes. In conclusion, all three method in Response class is responsible for the same work, so this class has high cohesion (**functional cohesion**).
* VNpayScreen class: besides constructors, this class’ main purpose is displaying the web page through displayWeb() (supported by handleUrlChange), so this class is **functional cohesion**.
* VNPayConfig class: this class has multiple methods with different purpose and these purpose only connect logically (all related about provide services for the package) so this is **logical cohesion**. To avoid this issue, we should move each method with different responsiblity to different classes.

Code modification: all new classes is moved to package subsystem.vnpay.utilities

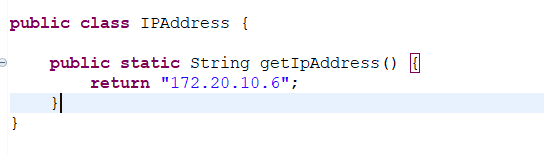
* New SHA256Generate class:



* New MD5Generate class:



* New IPAddress class:



* New RandomNumberGenerate class:

