|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module No:** | 1 | **IU No:** | 1 | **Exercise No.** | 1 |

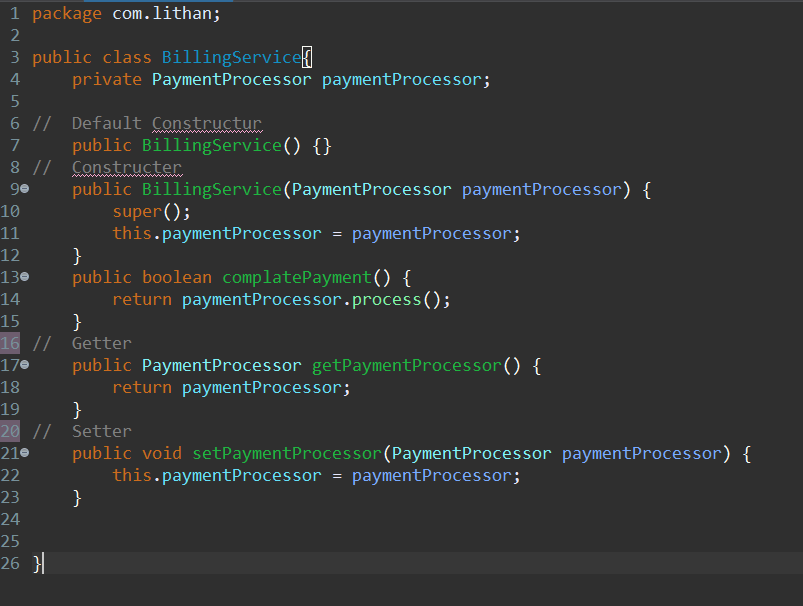
|  |  |
| --- | --- |
| **Lab Assessment Statement** | **Assignment 1 - Inversion of Control**  **You are handling the Billing module of an e-commerce application. You have developed below class.**  package lithan.training.javawebapp  public class BillingService {  private PaymentProcessor paymentProcessor;    public bool completePayment() {  return paymentProcessor.process();  }  }  **Note:**   * **You are using a 3rd party library (external library) to process the payments** * **This external library provides an interface PaymentProcessor for clients to process the payments** * **The external library supports 2 modes of payment. “gpay” and “credit\_card”.** * **It also has a Factory class which will instantiate appropriate processor based on mode of payment.** * **Below is the brief outline of the classes provided by external library**   public interface PaymentProcessor {  public bool process();  }  public class GooglePayProcessor implements PaymentProcessor {  public bool process() {  // process payment  return true;  }  }  public class CreditCardProcessor implements PaymentProcessor {  public bool process() {  // process payment  return true;  }  }  public PaymentProcessorFactory {  private final GooglePayProcessor gpayProcessor = new GooglePayProcessor();  private final CreditCardProcessor cardProcessor = new CreditCardProcessor ();    public PaymentProcessor getPaymentProcessor(String mode) {  if ("gpay".equalsIgnoreCase(mode)) {  return gpayProcessor;  } else if ("credit\_card".equalsIgnoreCase(mode)) {  return cardProcessor ;  }  return null;  }  }     1. **Enhance the BillingService class so it gets instantiated and the dependencies shall get injected.**    * Hint: What method would you use to instantiate BillingService class? Remember, this is the bean developed by you. 2. **Write Java Configuration class to instantiate required classes from external library.** |
| **Technical Environment** | - |
| **Guidelines** | - |
| **Duration** | 120 mins |

Solution

* + - 1. **Enhance the BillingService class so it gets instantiated and the dependencies shall get injected.**

Adding the constructor and getter setter method would allow the Billing Service to get Instantiated ang dependency to be injected.

BillingService.java



* + - 1. **Write Java Configuration class to instantiate required classes from external library.**

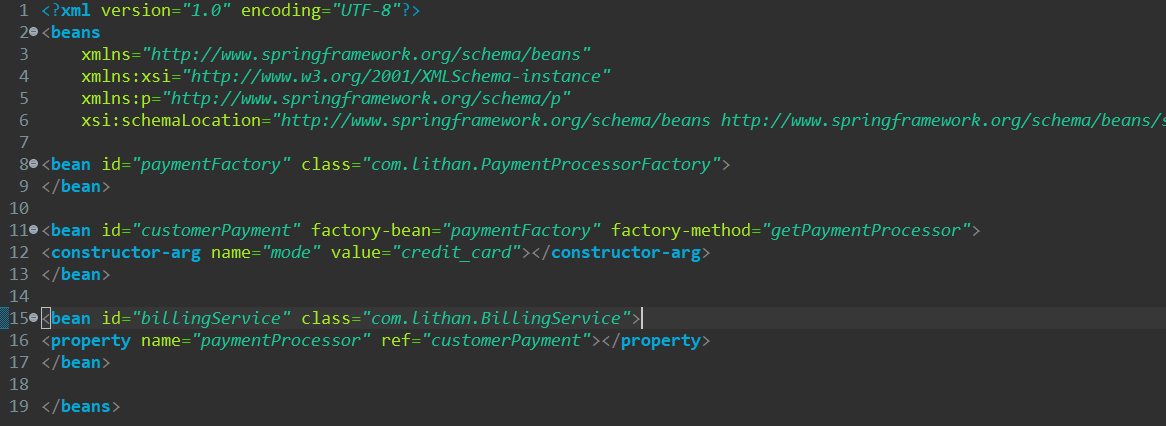
Before using bean, ApplicationContext is needed to read the config file, that’s why Action context need to be created. to use the bean, first bean service object need to be creates by casting context and get bean by their id, and the bean is successfully created.

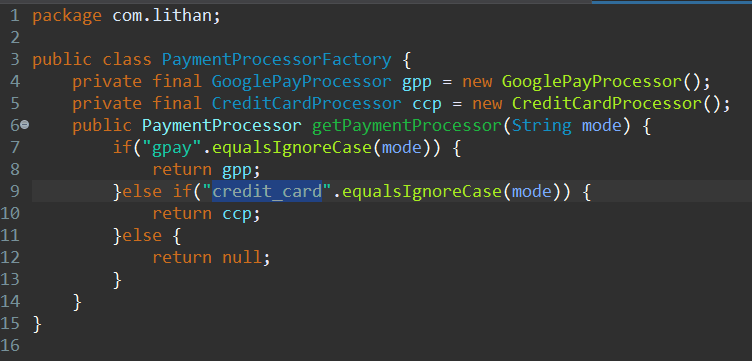
App.java



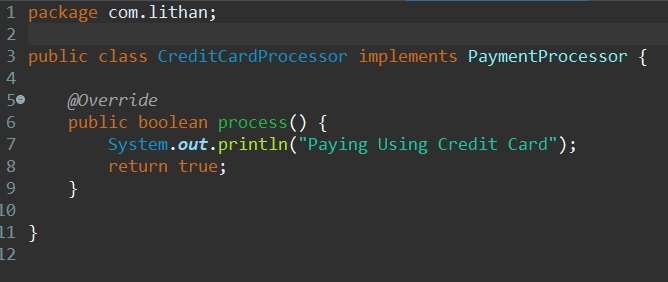
The used id in App.java is on line no 15, by defining the property paymentProccess. The value that is sanded to paymentProcess is referred to customerPayment which is paymentFactory, the object will be created based the value of the method in the constructor. If it gpay the object will be GooglePay and if it credit\_card the object will be CreditCard, this is called the factory method in design pattern.

Config.xml

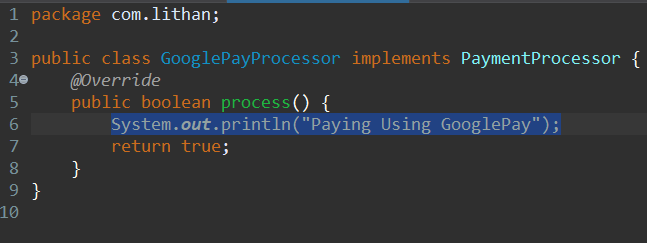


PaymentProcessorFactory.java

CreditCardProcessor.java



GooglePayProcessor.java



**Result:**

