Lesson 2 Configuring Charge Invoicing Behaviors

2.1 Bill an item immediately

Succeed Insurance wants to customize when deposits are billed based on the policy issuance date.

2.1.1 Requirements

Spec 1 If the effective date of a policy issuance billing instruction is earlier than today, then bill the down payment immediately.

Spec 2 Not applicable to Agency Bill processing

2.1.2 Tasks

1. Configure the customizeChargeInitializer() method in ChargeInitializer plugin.
2. Compile your changes

2.1.3 Testing procedure

1. Create a new account
    a. Use quick jump to execute the Run Account command
2. Add a policy to the new account with these details: (Actions → Add Policy)

a.	Policy Number:	BillToday 1
b.	Effective Date:	5 days earlier than the BillingCenter clock date
c.	Expiration Date:	1 year after the effective date
d.	Payment Plan:	PP03
e.	Premium:	$2500
f.	Taxes:	$220

3. Go to the Invoice screen (Account → Invoices) and verify that:
    a. The premium down payment is invoiced on the same day as the BillingCenter clock date.
    b. The installment premium and taxes are charged on the second invoice (that is, the first regularly scheduled invoice)
4. Add a policy to the new account with these details: (Actions → Add Policy)

a.	Policy Number:	BillToday 2
b.	Effective Date:	5 days after than the BillingCenter clock date
c.	Expiration Date:	1 year after the effective date
d.	Payment Plan:	PP03
e.	Premium:	$600
f.	Taxes:	$45

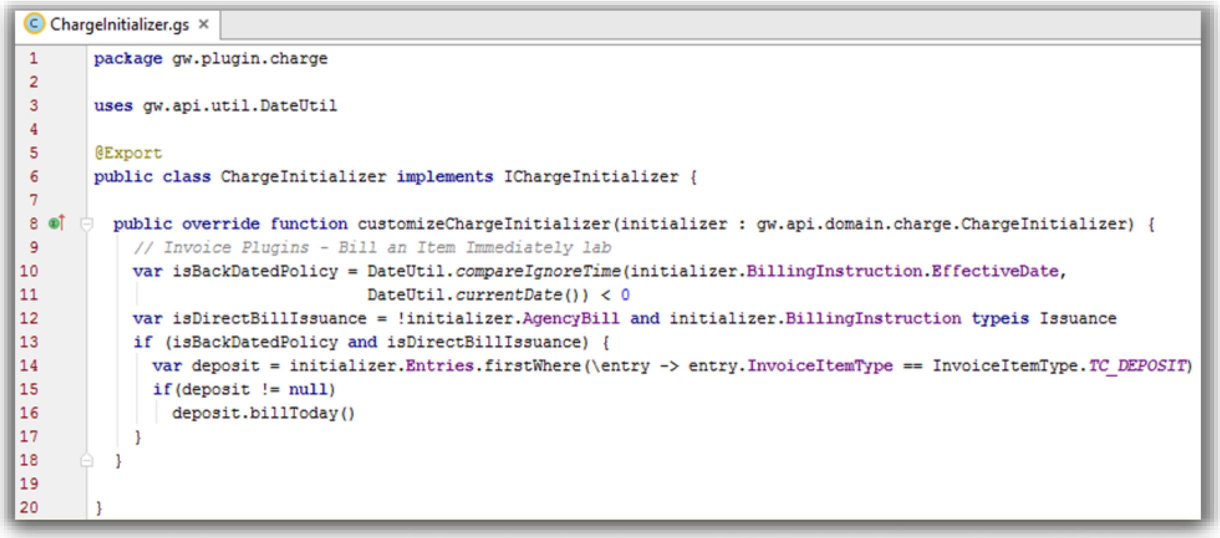
5. Go to the Invoice screen (Account → Invoices) and verify that:
    a. Down payment and taxes are on the same invoice.

2.1.4 Solution

Solution

Exact details on how to complete the exercise.

1. Configure the customizeChargeInitializer() method in ChargeInitializer plugin.



```
package gw.plugin.charge

uses gw.api.util.DateUtil

@Export
public class ChargeInitializer implements IChargeInitializer {

  public override function customizeChargeInitializer(initializer : gw.api.domain.charge.ChargeInitializer) {
    // Invoice Plugins - Bill an Item Immediately lab
    var isBackDatedPolicy = DateUtil.compareIgnoreTime(initializer.BillingInstruction.EffectiveDate,
                            DateUtil.currentDate()) < 0
    var isDirectBillIssuance = !initializer.AgencyBill and initializer.BillingInstruction typeis Issuance
    if (isBackDatedPolicy and isDirectBillIssuance) {
      var deposit = initializer.Entries.firstWhere(\entry -> entry.InvoiceItemType == InvoiceItemType.TC_DEPOSIT)
      if(deposit != null)
        deposit.billToday()
    }
  }

}
```

```
package gw.plugin.charge

uses gw.api.util.DateUtil

@Export
public class ChargeInitializer implements IChargeInitializer {

  public override function customizeChargeInitializer(initializer :
gw.api.domain.charge.ChargeInitializer) {
    // Invoice Plugins - Bill an Item Immediately lab
    var isBackDatedPolicy = DateUtil.compareIgnoreTime(initializer.BillingInstruction.EffectiveDate,
                            DateUtil.currentDate()) < 0
    var isDirectBillIssuance = !initializer.AgencyBill and initializer.BillingInstruction typeis
Issuance
    if (isBackDatedPolicy and isDirectBillIssuance) {
      var deposit = initializer.Entries.firstWhere(\entry -> entry.InvoiceItemType ==
InvoiceItemType.TC_DEPOSIT)
      if(deposit != null)
        deposit.billToday()
    }
  }

}
```

2. Compile your changes

a. Execute Run → Debugging Actions → Reload Changed Classes

2.2 Demo code: Splitting the down payment

The following code is from the instructor demo:

```
package gw.plugin.charge
@Export
public class ChargeInitializer implements IChargeInitializer {

  public override function customizeChargeInitializer(initializer :
gw.api.domain.charge.ChargeInitializer) {
    //Insert Code here to modify the ChargeInitializer's Entries that will be used for creation and
placement of InvoiceItems.

    var isDirectBillIssuance = !initializer.AgencyBill and initializer.BillingInstruction typeis
Issuance
    if(isDirectBillIssuance) {
      var paymentPlan = initializer.PolicyPeriod.PaymentPlan
      if(paymentPlan.SplitDownPayment_Ext
          and paymentPlan.DownPaymentPercent >= 20
          and paymentPlan.Periodicity == Periodicity.TC_MONTHLY) {
        // find amounts
        var firstDownPaymentItem = initializer.Entries.firstWhere(\entry -> entry.InvoiceItemType ==
InvoiceItemType.TC_DEPOSIT)
        var totalDownPaymentAmount = firstDownPaymentItem.Amount
        var newFirstDownPaymentAmount = totalDownPaymentAmount / 2
        var newSecondDownPaymentAmount = totalDownPaymentAmount - newFirstDownPaymentAmount
        var newSecondDownEventDate = firstDownPaymentItem.EventDate.addMonths(1)
        // set new amounts
        firstDownPaymentItem.Amount = newFirstDownPaymentAmount
        initializer.addEntry(newSecondDownPaymentAmount, InvoiceItemType.TC_DEPOSIT,
newSecondDownEventDate)
      }
    }
  }
}
```