Lesson 3 Configuring Invoice Streams

3.1 Implement a new payment interval

Succeed Insurance wants the ability to invoice some customers every five months.

3.1.1 Requirements

Spec 1 Policies are invoiced every five months.

3.1.2 Tasks

1. Add *everyfivemonths* to Periodicity typelist.
   a. Restart server to deploy database changes.
2. Configure createPeriodicSequenceWith() method in *DateSequence* plugin to add new periodicity.
3. Configure getInvoiceStreamPeriodicityFor() method in *InvoiceStream* plugin.
   a. Compile code changes

3.1.3 Testing procedure

1. Clone the PP03 payment plan. *(Administration → Business Settings → Payment Plans)*

a.	Name:	Every Five Months Plan
b.	Payment Interval:	Every Five Months

2. Create an account. (QuickJump: Run Account)
3. Add a policy to the new account with these details: *(Actions → Add Policy)*

a.	Policy Number:	EFM
b.	Expiration Date:	2-years after effective date
c.	Payment Plan:	Every Five Months Plan
d.	Premium:	$1300

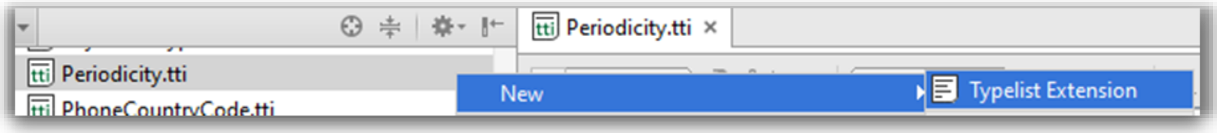
4. Go to the Invoice screen (Account → Invoices) and verify that:
   a. The monthly invoice stream was used for the invoices.
   b. The invoice periodicity is *every five months*.

3.1.4 Solution

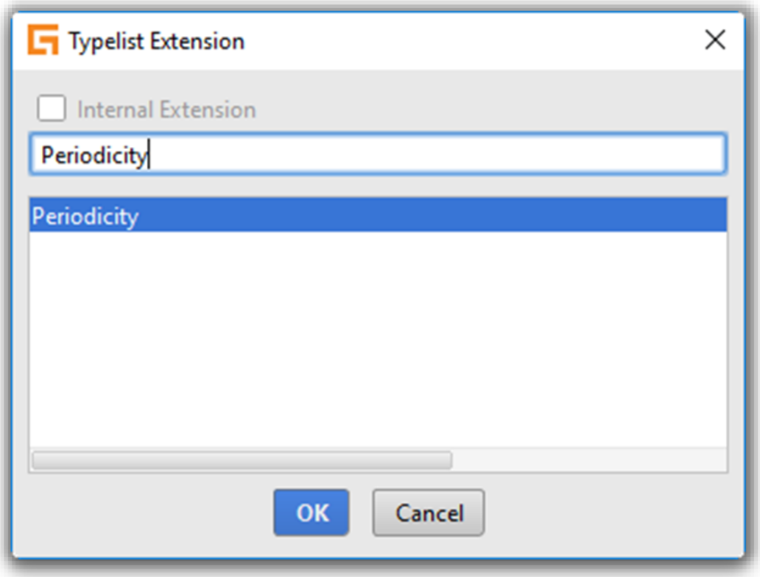
Solution

Exact details on how to complete the exercise.

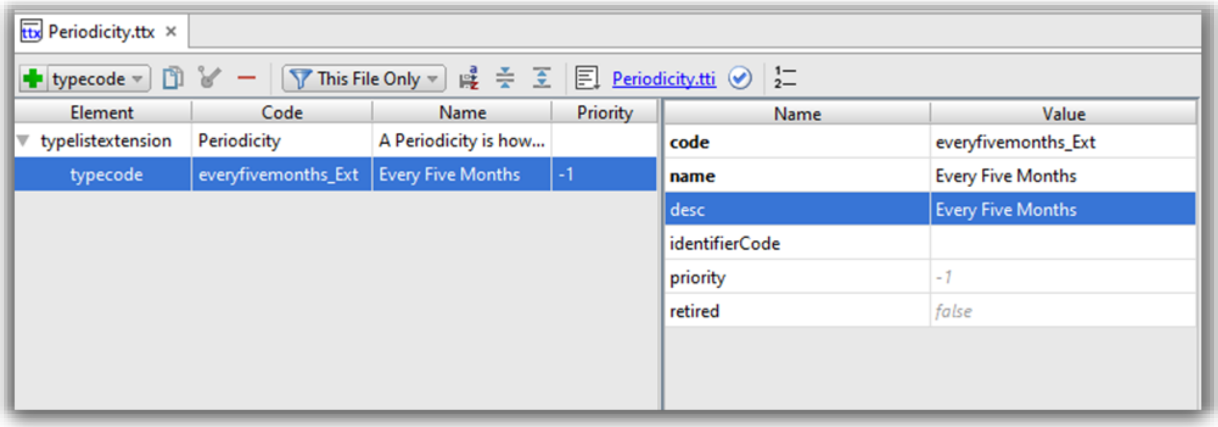
1. Add everyfivemonths to Periodicity typelist.
    a. In Studio, go to configuration → config → Extensions → Typelist
    b. Right click on Typelist, select New → Typelist Extension



    c. In the Typelist Extension window, enter “Periodicity” and then click the OK button.



    d. Add new typecode value

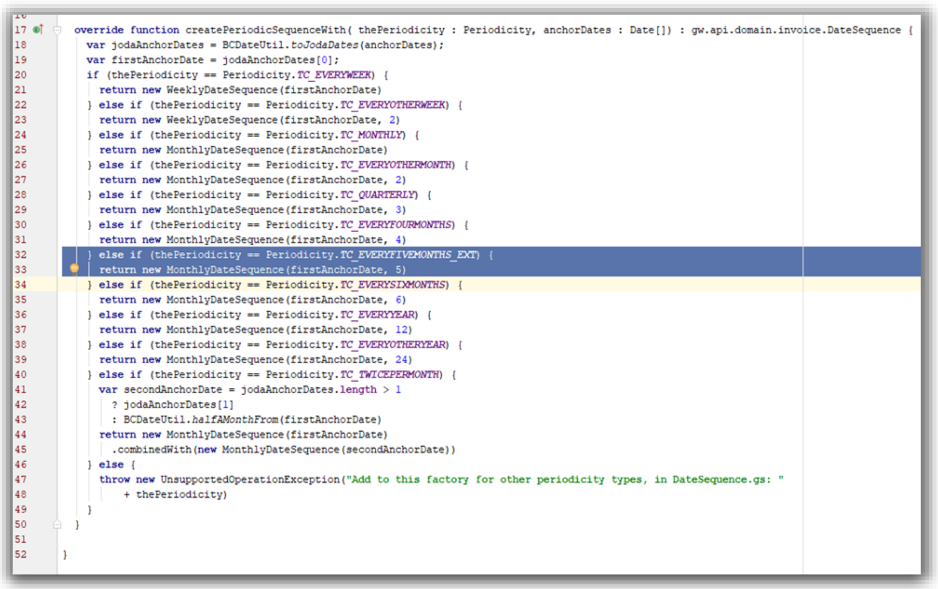


Element	Code	Name	Priority
typelistextension	Periodicity	A Periodicity is how...	
typecode	everyfivemonths_Ext	Every Five Months	-1

Name	Value
code	everyfivemonths_Ext
name	Every Five Months
desc	Every Five Months
identifierCode	
priority	-1
retired	false

    e. Restart server to deploy database changes.

2. Configure createPeriodicSequenceWith() method in *DateSequence* plugin to add new periodicity.



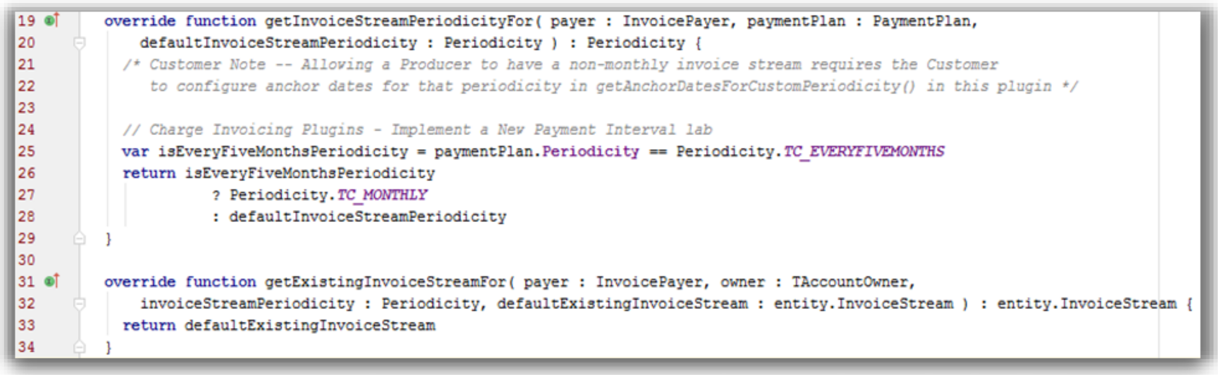
```
override function createPeriodicSequenceWith( thePeriodicity : Periodicity, anchorDates : Date[]) : gw.api.domain.invoice.DateSequence {
  var jodaAnchorDates = BCDateUtil.toJodaDates(anchorDates);
  var firstAnchorDate = jodaAnchorDates[0];
  if (thePeriodicity == Periodicity.TC_EVERYWEEK) {
    return new WeeklyDateSequence(firstAnchorDate)
  } else if (thePeriodicity == Periodicity.TC_EVERYOTHERWEEK) {
    return new WeeklyDateSequence(firstAnchorDate, 2)
  } else if (thePeriodicity == Periodicity.TC_MONTHLY) {
    return new MonthlyDateSequence(firstAnchorDate)
  } else if (thePeriodicity == Periodicity.TC_EVERYOTHERMONTH) {
    return new MonthlyDateSequence(firstAnchorDate, 2)
  } else if (thePeriodicity == Periodicity.TC_QUARTERLY) {
    return new MonthlyDateSequence(firstAnchorDate, 3)
  } else if (thePeriodicity == Periodicity.TC_EVERYFOURMONTHS) {
    return new MonthlyDateSequence(firstAnchorDate, 4)
  } else if (thePeriodicity == Periodicity.TC_EVERYFIVEMONTHS_EXT) {
    return new MonthlyDateSequence(firstAnchorDate, 5)
  } else if (thePeriodicity == Periodicity.TC_EVERYSIXMONTHS) {
    return new MonthlyDateSequence(firstAnchorDate, 6)
  } else if (thePeriodicity == Periodicity.TC_EVERYYEAR) {
    return new MonthlyDateSequence(firstAnchorDate, 12)
  } else if (thePeriodicity == Periodicity.TC_EVERYOTHERYEAR) {
    return new MonthlyDateSequence(firstAnchorDate, 24)
  } else if (thePeriodicity == Periodicity.TC_TWICEPERMONTH) {
    var secondAnchorDate = jodaAnchorDates.length > 1
      ? jodaAnchorDates[1]
      : BCDateUtil.halfAMonthFrom(firstAnchorDate)
    return new MonthlyDateSequence(firstAnchorDate)
      .combinedWith(new MonthlyDateSequence(secondAnchorDate))
  } else {
    throw new UnsupportedOperationException("Add to this factory for other periodicity types, in DateSequence.gs: "
        + thePeriodicity)
  }
}

}
```

```
      } else if (thePeriodicity == Periodicity.TC_EVERYFIVEMONTHS_EXT) {
  return new MonthlyDateSequence(firstAnchorDate, 5)
```

3. Configure getInvoiceStreamPeriodicityFor() method in *InvoiceStream* plugin.

   a. Update the code as shown below



```
override function getInvoiceStreamPeriodicityFor( payer : InvoicePayer, paymentPlan : PaymentPlan,
    defaultInvoiceStreamPeriodicity : Periodicity ) : Periodicity {
  /* Customer Note -- Allowing a Producer to have a non-monthly invoice stream requires the Customer
     to configure anchor dates for that periodicity in getAnchorDatesForCustomPeriodicity() in this plugin */

  // Charge Invoicing Plugins - Implement a New Payment Interval lab
  var isEveryFiveMonthsPeriodicity = paymentPlan.Periodicity == Periodicity.TC_EVERYFIVEMONTHS
  return isEveryFiveMonthsPeriodicity
          ? Periodicity.TC_MONTHLY
          : defaultInvoiceStreamPeriodicity
}

override function getExistingInvoiceStreamFor( payer : InvoicePayer, owner : TAccountOwner,
    invoiceStreamPeriodicity : Periodicity, defaultExistingInvoiceStream : entity.InvoiceStream ) : entity.InvoiceStream {
  return defaultExistingInvoiceStream
}
```

```
      override function getInvoiceStreamPeriodicityFor( payer : InvoicePayer, paymentPlan :
PaymentPlan,
    defaultInvoiceStreamPeriodicity : Periodicity ) : Periodicity {
  /* Customer Note -- Allowing a Producer to have a non-monthly invoice stream requires the Customer
     to configure anchor dates for that periodicity in getAnchorDatesForCustomPeriodicity() in this
plugin */

  // Charge Invoicing Plugins - Implement a New Payment Interval lab
  var isEveryFiveMonthsPeriodicity = paymentPlan.Periodicity == Periodicity.TC_EVERYFIVEMONTHS_EXT
  return isEveryFiveMonthsPeriodicity
          ? Periodicity.TC_MONTHLY
          : defaultInvoiceStreamPeriodicity
}
```

b. Compile code changes. *(Run → Debugging Actions → Reload Changed Classes)*

3.2 Demo code: Configure separate invoice stream for quarterly policies

The following code is from the instructor demo:

```
       override function getInvoiceStreamPeriodicityFor( payer : InvoicePayer, paymentPlan :
PaymentPlan,
    defaultInvoiceStreamPeriodicity : Periodicity ) : Periodicity {
  /* Customer Note -- Allowing a Producer to have a non-monthly invoice stream requires the Customer
     to configure anchor dates for that periodicity in getAnchorDatesForCustomPeriodicity() in this
plugin */

  // Invoice Stream Demo
  var isQuarterlyPeriodicity = paymentPlan.Periodicity == Periodicity.TC_QUARTERLY
  return isQuarterlyPeriodicity
            ? Periodicity.TC_QUARTERLY
            : defaultInvoiceStreamPeriodicity
}
       override function getAnchorDatesForCustomPeriodicity( invoicePayer : InvoicePayer,
customPeriodicity : Periodicity )
    : List<AnchorDate> {
  // Invoice Stream Demo
  var strDate = "01/05/2000"
  var dtf = DateTimeFormat.forPattern("MM/dd/YYYY") // format for input
  var jodaDate = dtf.parseDateTime(strDate)         // parsing the date

  return customPeriodicity == Periodicity.TC_QUARTERLY
            ? new ArrayList<AnchorDate>() {AnchorDate.fromDate(jodaDate)}  // year does not matter
            : new ArrayList<AnchorDate>()                                  // default return value
}
```