```
#created on: 10-sep-2009
package
ar.gob.anses.prissa.mi.asistente_reglas.prototipo.simulaci
on.reglas;

import
ar.gob.anses.prissa.mi.asistente_reglas.prototipo.simulaci
on.modelo.*;

rule "Fecha Cese"
    when
        #conditions
        //El Beneficiario no tiene fechaCese
        $Beneficiario : Beneficiario ( fechaInicialPago
== null )

        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene Fecha Cese"
        System.out.println("El Beneficiario no tiene
Fecha Inicial de Pago");

        //Agregar Rechazo "El Beneficiario no tiene Fecha
de Cese"
        $Beneficiario.getBeneficio().addRechazo("El
Beneficiario no tiene Fecha de Cese");
end


rule "Edad"
    when
        #conditions
```

```
        //El Beneficiario tiene mas de "65" anios desde
fechaNacimiento
        $Beneficiario : Beneficiario ( eval
( ValidationHelper.yearsPassedSince( fechaNacimiento ) >
65 ) )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "El Beneficiario no tiene menos de 65
anios"
        System.out.println("El Beneficiario no tiene
menos de 65 anios");

        //Agregar Rechazo "No tiene menos de 65 anios"
        $Beneficiario.getBeneficio().addRechazo("El
Beneficiario no tiene menos de 65 anios");
end


rule "Regularidad"
    when
        #conditions
        //El Beneficiario no tiene regularidad "REGULAR"
ni "IRREGULAR_CON_DERECHO"
        $Beneficiario : Beneficiario ( regularidad !=
Beneficiario.TipoRegularidad.REGULAR && !=
Beneficiario.TipoRegularidad.IRREGULAR_CON_DERECHO )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene la condicion de regularidad
```

```
necesaria"
        System.out.println("No tiene la condicion de
regularidad necesaria");

        //Agregar Rechazo "No tiene la condicion de
regularidad necesaria"
        $Beneficiario.getBeneficio().addRechazo("No tiene
la condicion de regularidad necesaria");
end


rule "Informe Medico"
    when
        #conditions
        //El Beneficiario tiene resultado de informe
medico menor a "66"
        $Beneficiario : Beneficiario
( resultadoInformeMedico < 66 )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene el porcentaje de invalidez
necesario"
        System.out.println("No tiene el porcentaje de
invalidez necesario");

        //Agregar Rechazo "No tiene el porcentaje de
invalidez necesario"
        $Beneficiario.getBeneficio().addRechazo("No tiene
el porcentaje de invalidez necesario");
end
```

```
rule "Acreditacion de Identidad"
    when
        #conditions
        //El Beneficiario no tiene valido
documentacionProbatoria
        $Beneficiario : Beneficiario
( identidadAcreditada == false )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene probatoria de servicios
presentada"
        System.out.println("No tiene acreditada su
identidad");

        //Agregar Rechazo "No tiene probatoria de
servicios presentada"
        $Beneficiario.getBeneficio().addRechazo("No tiene
correctamente acreditada su identidad");
end

rule "Incompatibilidad"
    when
        #conditions
        //El Beneficiario no tiene valido
documentacionProbatoria
        $Beneficiario : Beneficiario ( incompatibilidad
== true )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
```

```
        //Log : "No tiene probatoria de servicios
presentada"
        System.out.println("Esta cobrando algun otro
beneficio incompatible");

        //Agregar Rechazo "No tiene probatoria de
servicios presentada"
        $Beneficiario.getBeneficio().addRechazo("Esta
cobrando algun otro beneficio incompatible");
end

rule "Accidente Laboral"
    when
        #conditions
        //El Beneficiario no tiene valido
documentacionProbatoria
        $Beneficiario : Beneficiario ( accidenteLaboral
== true )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene probatoria de servicios
presentada"
        System.out.println("La causa de invalidez se debe
a accidente laboral");

        //Agregar Rechazo "No tiene probatoria de
servicios presentada"
        $Beneficiario.getBeneficio().addRechazo("La causa
de invalidez se debe a accidente laboral");
end

rule "Sentencia Firme"
```

```
    when
        #conditions
        //El Beneficiario no tiene valido
documentacionProbatoria
        $Beneficiario : Beneficiario ( sentenciaFirme ==
false )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene probatoria de servicios
presentada"
        System.out.println("El informe medico aun no
posee sentencia firme");

        //Agregar Rechazo "No tiene probatoria de
servicios presentada"
        $Beneficiario.getBeneficio().addRechazo("El
informe medico aun no posee sentencia firme");
end

rule "Trabajador Activo"
    when
        #conditions
        //El Beneficiario no tiene valido
documentacionProbatoria
        $Beneficiario : Beneficiario ( trabajadorActivo
== true )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene probatoria de servicios
```

```
presentada"
        System.out.println("El beneficiaro es un
trabajador activo");

        //Agregar Rechazo "No tiene probatoria de
servicios presentada"
        $Beneficiario.getBeneficio().addRechazo("El
beneficiaro es un trabajador activo");
end

rule "Ingreso Base"
    when
        #conditions
        //El Beneficiario tiene ingresoBase en "0"
        $Beneficiario : Beneficiario ( ingresoBase == 0 )
        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)
    then
        #actions
        //Log : "No tiene calculado el ingreso base"
        System.out.println("No tiene calculado el ingreso
base");

        //Agregar Rechazo "No tiene calculado el ingreso
base"
        $Beneficiario.getBeneficio().addRechazo("No tiene
calculado el ingreso base");
end


rule "Tipo de Trabajdor"
    when
        #conditions
        //El Beneficiario no es relacionDependencia ni
```

```
autonomo
        $Beneficiario : Beneficiario
( relacionDependencia == false && autonomo == false )

        //El estado del beneficio es INVALIDO
        eval( $Beneficiario.getBeneficio().getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO)

    then
        #actions
        //Log : "No tiene configurada la condicion de
contribuyente en autonomo o dependencia"
        System.out.println("No tiene configurada la
condicion de tipo de trabajador en autonomo o
dependencia");

        //Agregar Rechazo "No tiene configurada la
condicion de contribuyente en autonomo o dependencia"
        $Beneficiario.getBeneficio().addRechazo("No tiene
configurada la condicion de tipo de trabajador en
autonomo o dependencia");
end

rule "Otorgamiento"
    salience -2
    when
        #conditions
        //Para todo Beneficiario que contiene beneficio
        $Beneficiario : Beneficiario
( beneficio.existenRechazos == false,

beneficio.getEstado ==
Beneficio.EstadoOtorgamiento.INDEFINIDO )

        //El beneficio esta INDEFINIDO
```

```
        //eval( $beneficio.getEstado() ==
Beneficio.EstadoOtorgamiento.INDEFINIDO )

        //El beneficio no tiene rechazos
        //eval( $beneficio.getRechazos().size() < 1 )
    then
        #actions
        //Log : "Se acepta el beneficio porque cumple con
todas las condiciones"
        //System.out.println("Se acepta el beneficio
porque cumple con todas las condiciones");
        //System.out.println("La coleccion tiene:" +
$Beneficiario.getBeneficio().getRechazos().size());
        //Modificar Beneficio con estado "OTORGADO"
        $Beneficiario.getBeneficio().setEstado
(Beneficio.EstadoOtorgamiento.OTORGADO);

        //Actualizar Beneficiario
        update($Beneficiario);
end


rule "No Otorgamiento"
    salience -1
    when
        #conditions
        //Para todo Beneficiario que contiene beneficio
        $Beneficiario : Beneficiario
( beneficio.isExistenRechazos == true,

beneficio.getEstado ==
Beneficio.EstadoOtorgamiento.INDEFINIDO )
```

```
    then
        #actions
        //Log : "Se rechaza el beneficio porque no cumple
con todas las condiciones"
        //System.out.println("Se rechaza el beneficio
porque no cumple con todas las condiciones");

        //Modificar Beneficio con estado "RECHAZADO"
        $Beneficiario.getBeneficio().setEstado
(Beneficio.EstadoOtorgamiento.RECHAZADO);

        //Actualizar Beneficiario
        update($Beneficiario);
end


rule "Calculo Regular"
    salience -1
    when
        #conditions
        //Para todo Beneficiario "REGULAR" con beneficio
        $Beneficiario : Beneficiario ( regularidad ==
Beneficiario.TipoRegularidad.REGULAR && $beneficio :
beneficio )


        //El beneficio esta OTORGADO
        eval( $beneficio.getEstado() ==
Beneficio.EstadoOtorgamiento.OTORGADO )
    then
        #actions
        //Log : "Se calcula el Haber Mensual para
Beneficiario Regular"
        //System.out.println("Se calcula el Haber Mensual
para Beneficiario Regular");
```

```
        //Calcular haberMensual con "0.7"
        $Beneficiario.getBeneficio().setHaberMensual(0.7
* $Beneficiario.getIngresoBase());
end


rule "Calculo Irregular con Derecho"
    salience -1
    when
        #conditions
        //Para todo Beneficiario "IRREGULAR_CON_DERECHO"
con beneficio
        $Beneficiario : Beneficiario ( regularidad ==
Beneficiario.TipoRegularidad.IRREGULAR_CON_DERECHO &&
$beneficio : beneficio )

        //El beneficio esta "OTORGADO"
        eval($beneficio.getEstado() ==
Beneficio.EstadoOtorgamiento.OTORGADO)
    then
        #actions
        //Log : "Se calcula el Haber Mensual para
Beneficiario Irregular con Derecho"
        //System.out.println("Se calcula el Haber Mensual
para Beneficiario Regular");

        //Calcular haberMensual con "0.5"
        $Beneficiario.getBeneficio().setHaberMensual(0.5
* $Beneficiario.getIngresoBase());
end
```