# Modelos de Software

Ingeniería en Sistemas de Información
2018

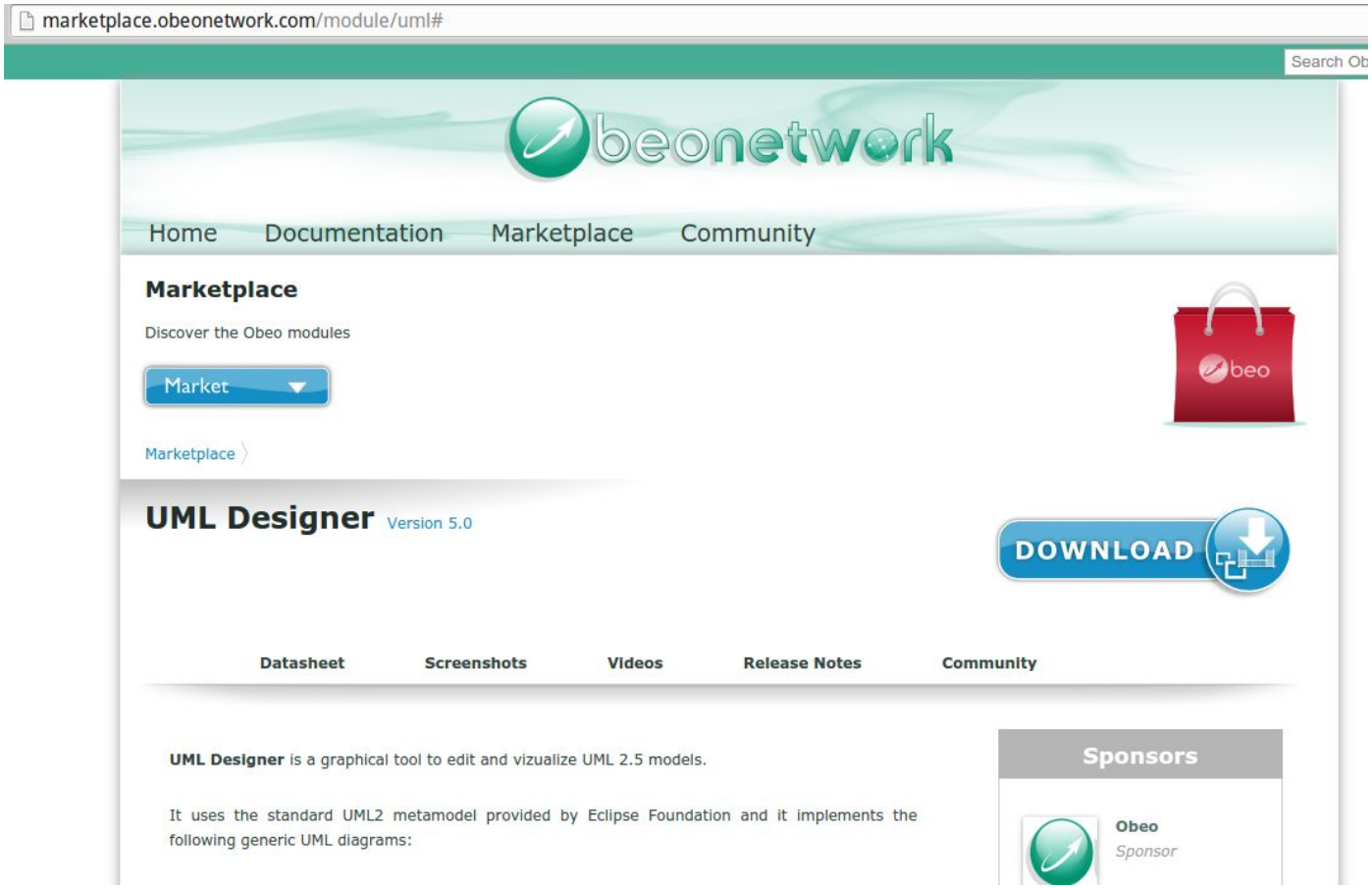
# Modelos de Software

## UML

La línea que separa el modelado de la programación a veces se vuelve algo difusa. Existen muchas herramientas que permiten transformar un diagrama de clases UML a código (y viceversa).

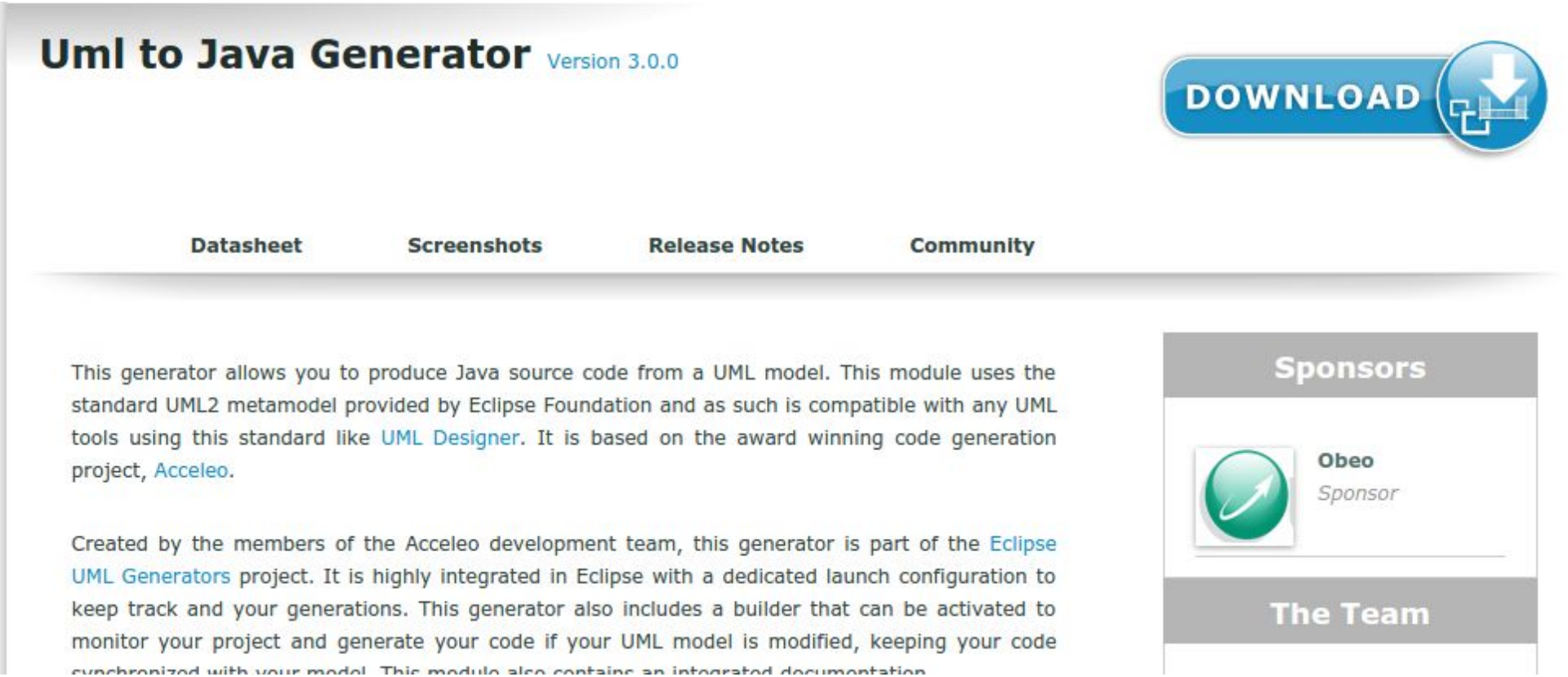
# Modelos de Software

UML to Java



# Modelos de Software

## UML to Java



# Modelos de Software

UML

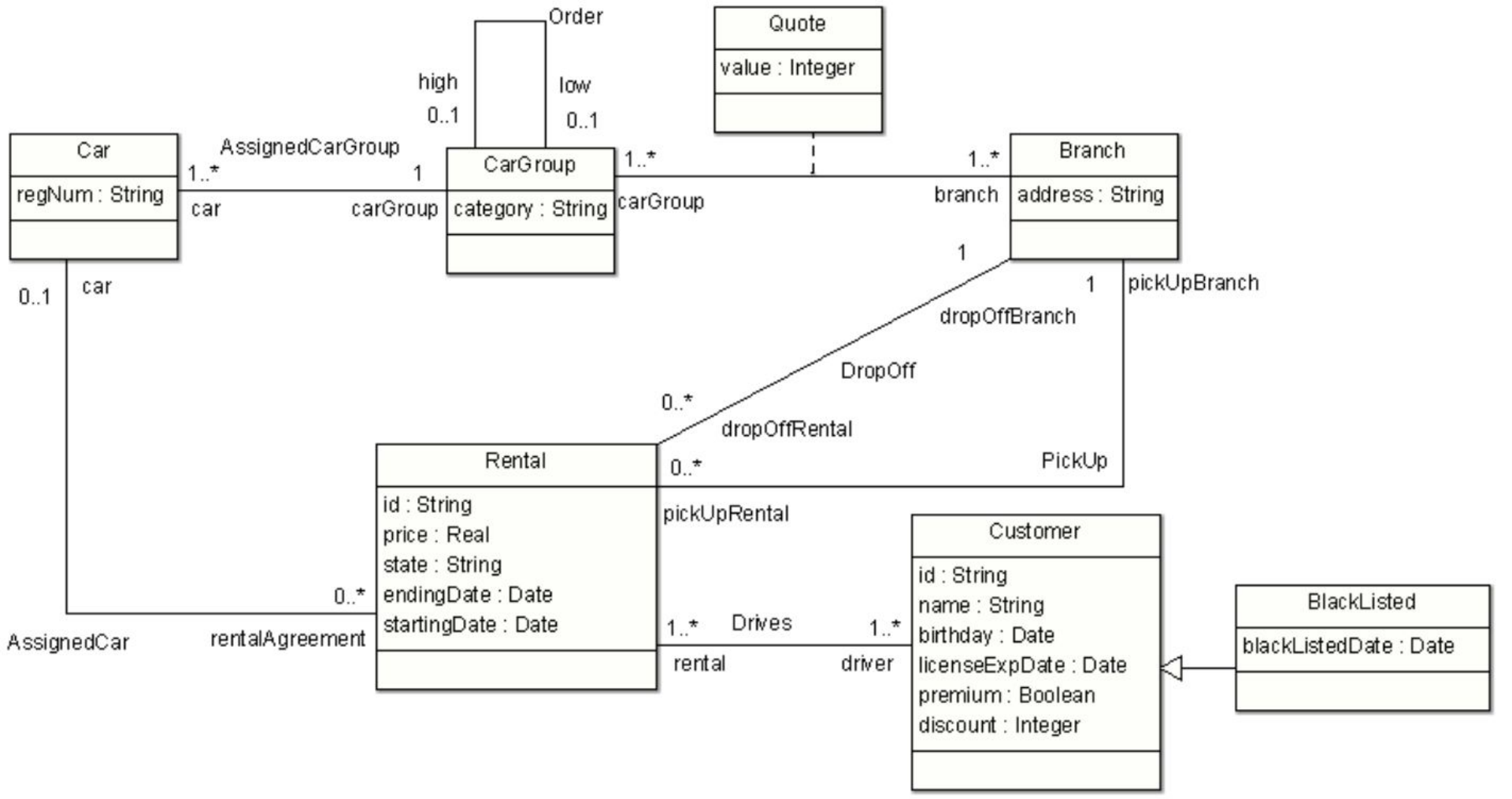
¿Qué otras cosas se pueden hacer con UML?

# Modelos de Software

## Limitaciones de UML

Para que un lenguaje de modelado sea claro y legible se debe limitar su expresividad. Esto significa que una representación gráfica, como es el diagrama de clase, sólo presenta un subconjunto de todo lo que es importante.

# Modelos de Software



# Modelos de Software

UML

Aunque este diagrama parece completo, hay cuestiones del sistema que no se están reflejando y son propias del problema:

- ¿Cómo se calcula el precio de un alquiler?
- ¿Puede una persona en una black list volver a alquilar?
- ¿Los lugares de pick up y drop off pueden ser diferentes?

# Modelos de Software

UML

Existen lenguajes que permiten introducir expresiones que complementan el modelado. Escribir notas en castellano junto al diagrama es una forma de complementar. Hoy vamos a ver metodologías más formales.

# Modelos de Software

Object Constraint Language (OCL)

# Modelos de Software

## UML + OCL

OCL es un lenguaje de especificación tipificado, declarativo y libre de side-effects.

*De Especificación: En OCL no describimos cómo se implementan las cosas.*

*Tipificado: Cada expresión en OCL evalúa a un tipo, y se puede operar en función de los tipos definidos.*

*Declarativo: No hay expresiones imperativos, como la asignación.*

*Libre de side-effects: Lo expresado en OCL no puede modificar el sistema.*

# Modelos de Software

## UML + OCL

Podemos utilizar OCL para

- Definición de invariantes
- Inicialización de propiedades de una clase
- Definición de reglas sobre cómo se calculan valores
- Definición de contratos

# Modelos de Software

## OCL - Invariantes

Los invariantes son restricciones de integridad sobre el sistema. Un invariante es, en esencia, una condición booleana que siempre debe ser verdadera.

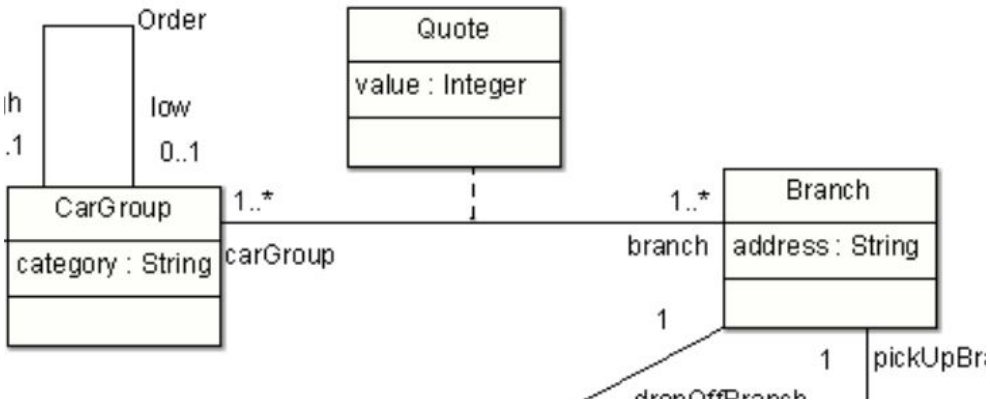
# Modelos de Software

## OCL - Invariantes

Los invariantes son restricciones de integridad sobre el sistema. Un invariante es, en esencia, una condición booleana que siempre debe ser verdadera.

Por ejemplo

```
context Quote inv QuoteOverZerp: self.value > 0
```



# Modelos de Software

## OCL - Invariantes

Los invariantes son restricciones de integridad sobre el sistema. Un invariante es, en esencia, una condición booleana que siempre debe ser verdadera.

Por ejemplo

```
context Quote inv QuoteOverZerp: self.value > 0
```

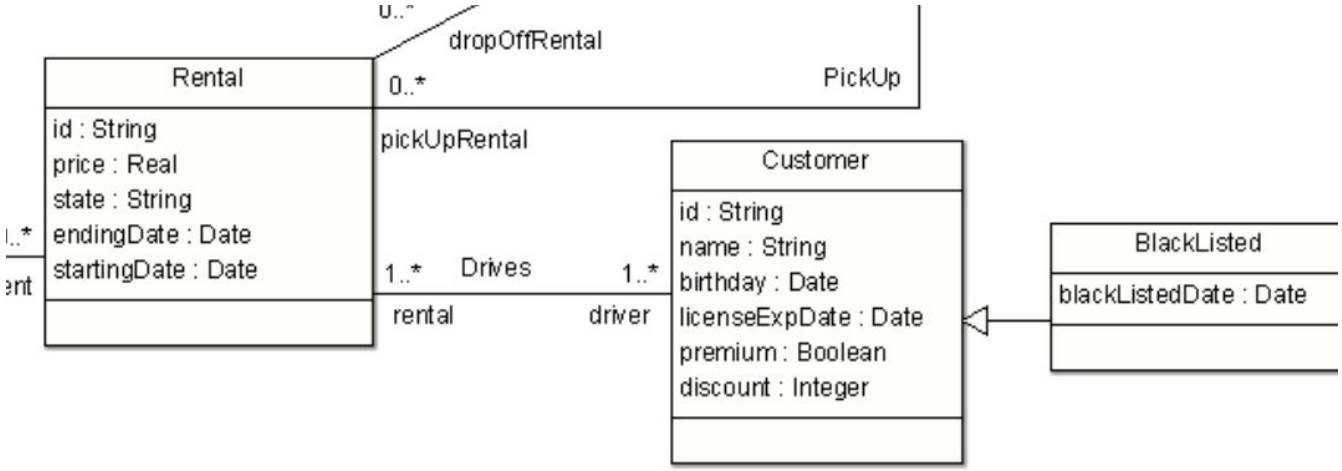
Todas las instancias de la clase Quote deben mantener esa condición booleana como verdadera.

# Modelos de Software

## OCL - Invariantes

Los invariantes pueden describir condiciones mucho más complejas

```
context BlackList inv NoRentalsBlackListed:
self.rental->forAll( r | r.startDate < self.blackListedDate )
```

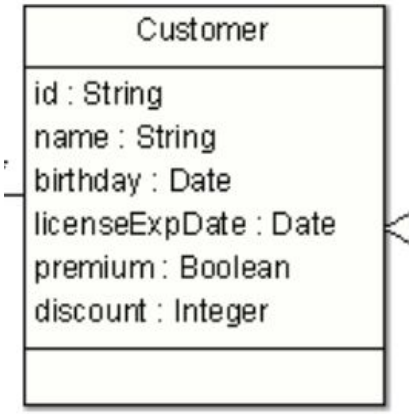


# Modelos de Software

## OCL - Inicialización

OCL se puede utilizar para definir cómo se deben inicializar los atributos de una clase cuando se crea una instancia de la misma.

```
context Customer::premium: boolean init: false
```

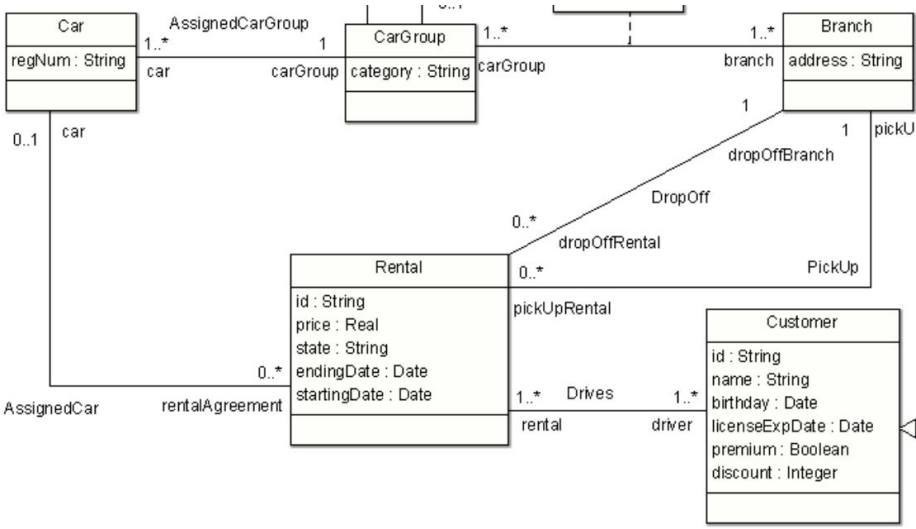


# Modelos de Software

## OCL - Elementos derivados

En OCL, un elemento derivado es aquel cuyo valor puede inferirse a partir del valor de otros elementos dentro del modelo.

```
context Customer::discount: integer
derive:
    if not self.premium then
        if self.rental.car.carGroup->
            select(c | c.category='high')
                ->size()>=5
        then 15
        else 0 endif
    else 30 endif
```

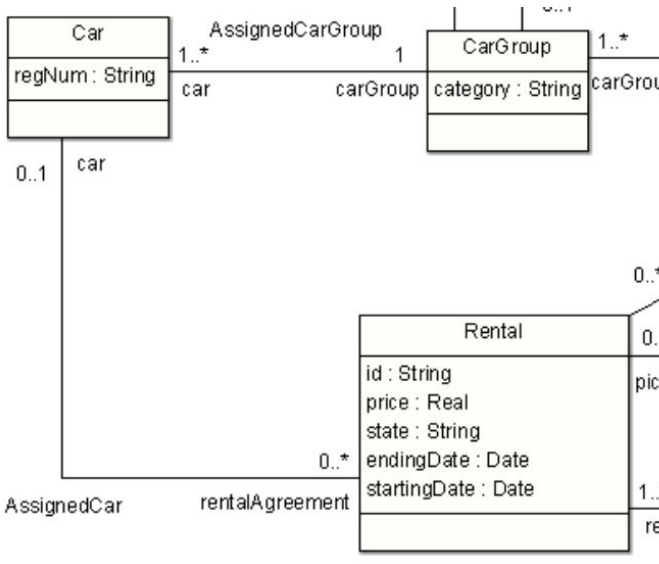


# Modelos de Software

## OCL - Consultas

En OCL podemos expresar consultas al modelo. Dichas consultas pueden ser utilizadas luego en otras expresiones de OCL.

```
context Car::mostPopular(): boolean
body: Car::allInstances()->forAll(c1 | c1<>self implies
      c1.rentalAgreement->size() <=
            self.rentalAgreement->size() )
```



# Modelos de Software

## OCL - Contratos

Cada operación tiene asociada un contrato. Un contrato es un conjunto de condiciones booleanas dividido en dos, pre y post condiciones.

Las precondiciones definen las condiciones de verdad sobre los parámetros de entrada y el estado del sistema antes de ejecutar la operación.

# Modelos de Software

## OCL - Contratos

Cada operación tiene asociada un contrato. Un contrato es un conjunto de condiciones booleanas dividido en dos, pre y post condiciones.

Las postcondiciones definen las condiciones de verdad del sistema cuando la operación termina.

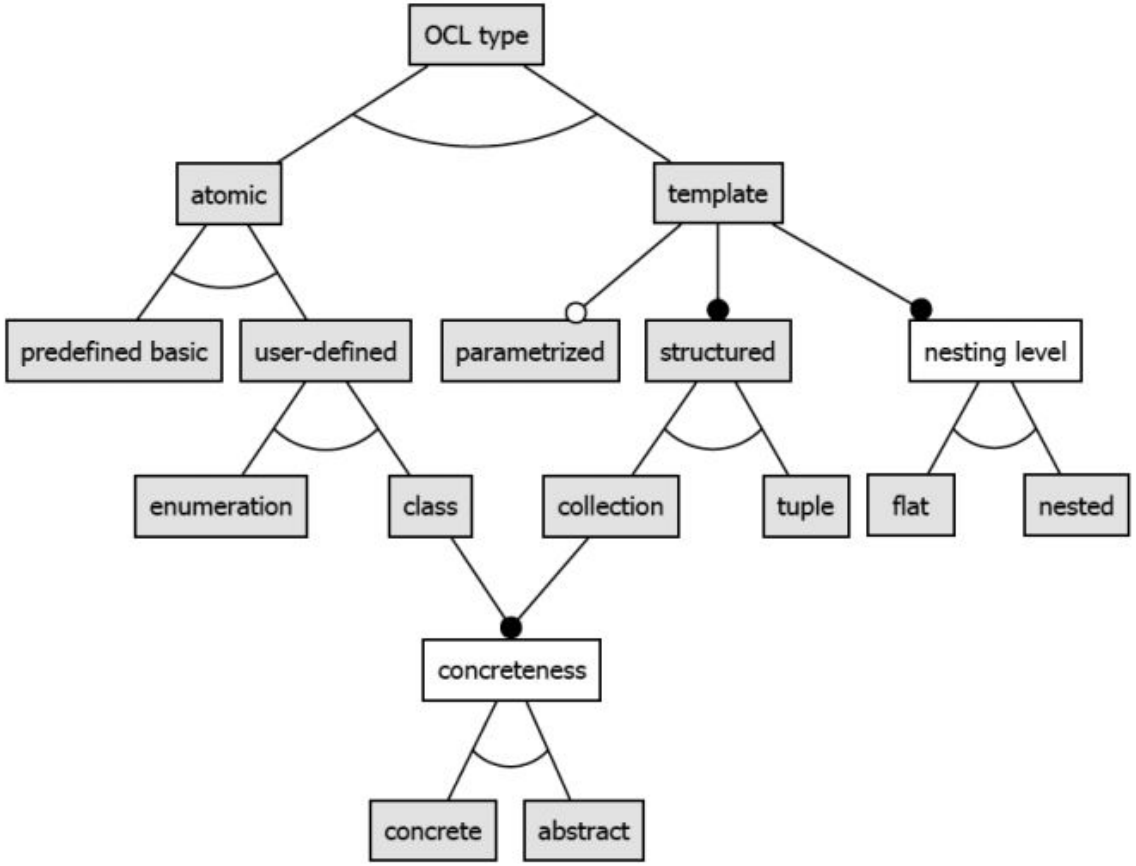
# Modelos de Software

## OCL - Contratos

```
context Rental::newRental(id:Integer, price:Real, startingDate:Date,
endingDate:Date, customer:Customer, carRegNum:String,
pickupBranch: Branch, dropOffBranch: Branch)
pre: customer.licenseExpDate>endingDate
post: Rental.allInstances->one(r |
    r.oclIsNew() and r.oclIsTypeOf(Rental) and
    r.endingDate=endingDate and r.startingDate=startingDate and
    r.driver=customer and r.pickupBranch=pickupBranch and
    r.dropOffBranch=dropOffBranch and
    r.car=Car.allInstances()->any(c | c.regNum=carRegNum))
```

# Modelos de Software

## OCL - Tipos



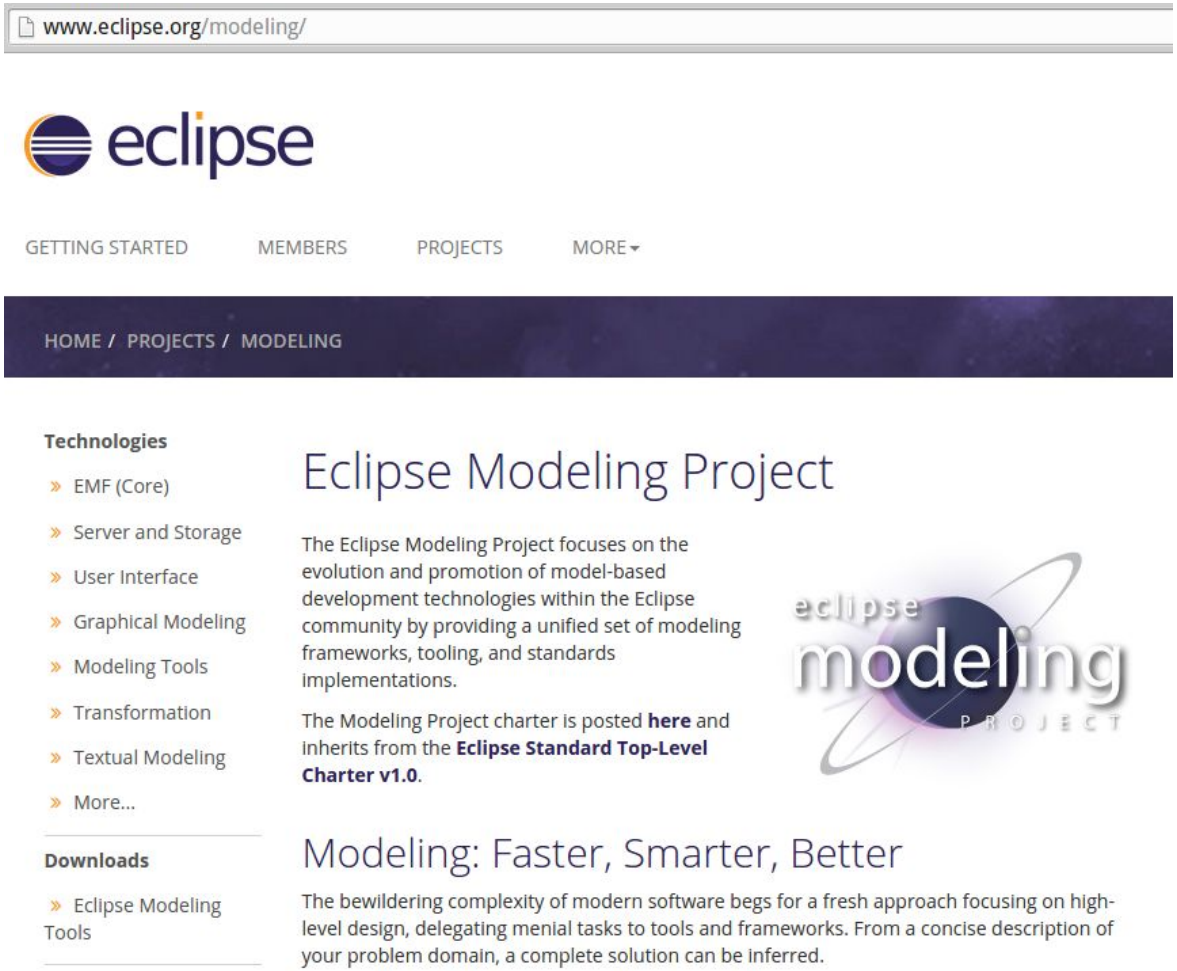
# Modelos de Software

## OCL - Eclipse

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/'

package ecore

context EModelElement
/**
 * Declare a helper operation to map an ok/warning verdict to ok/error.
*/
def: asError(verdict : Boolean) : Boolean = if verdict then true else null endif

/**
 * Extra validation for EReference and EAttribute.
 */
context EStructuralFeature

/**
 * Declare a helper property to determine whether an EStructuralFeature has an OCL derivation.
 */
def: hasDerivation : Boolean =
 eAnnotations->select(source.startsWith('http://www.eclipse.org/emf/2002/Ecore/OCL'))->notEmpty()

/**
 * If a feature has an OCL derivation, it should be transient; otherwise it is not executed.
 */
inv DerivationIsTransient: hasDerivation implies transient
```

# Modelos de Software

## OCL - Eclipse



# Modelos de Software

Java Modeling Language (JML)

# Modelos de Software

## Java Modeling Language (JML)

Al igual que OCL, JML permite declarar condiciones booleanas que el sistema debe respetar.

# Modelos de Software

Java Modeling Language (JML)

```
//@ requires x >= 0.0;
/*@ ensures JMLDouble
  @          .approximatelyEqualTo
  @           (x, \result * \result, eps);
  @*/
public static double sqrt(double x) {
  /*...*/
}
```

# Modelos de Software

Java Modeling Language (JML)

```
//@ requires x >= 0.0;
/*@ ensures JMLDouble
  @         .approximatelyEqualTo
  @          (x, \result * \result, eps);
  @*/
public static double sqrt(double x) {
  /*...*/
}
```

```
//@ requires (* x is positive *);
/*@ ensures (* \result is an
  @             approximation to
  @             the square root of x *)
  @      && \result >= 0;
  @*/
public static double sqrt(double x) {
  return Math.sqrt(x);
}
```

# Modelos de Software

Java Modeling Language (JML)

```
public class Person {
  private String name;
  private int weight;

  /*@ also
    @ ensures \result != null
    @   && (* \result is a displayable
    @       form of this person *);
    @*/
  public String toString() {
    return "Person(\"" + name + "\","
      + weight + ")";
  }
```

# Modelos de Software

Java Modeling Language (JML)

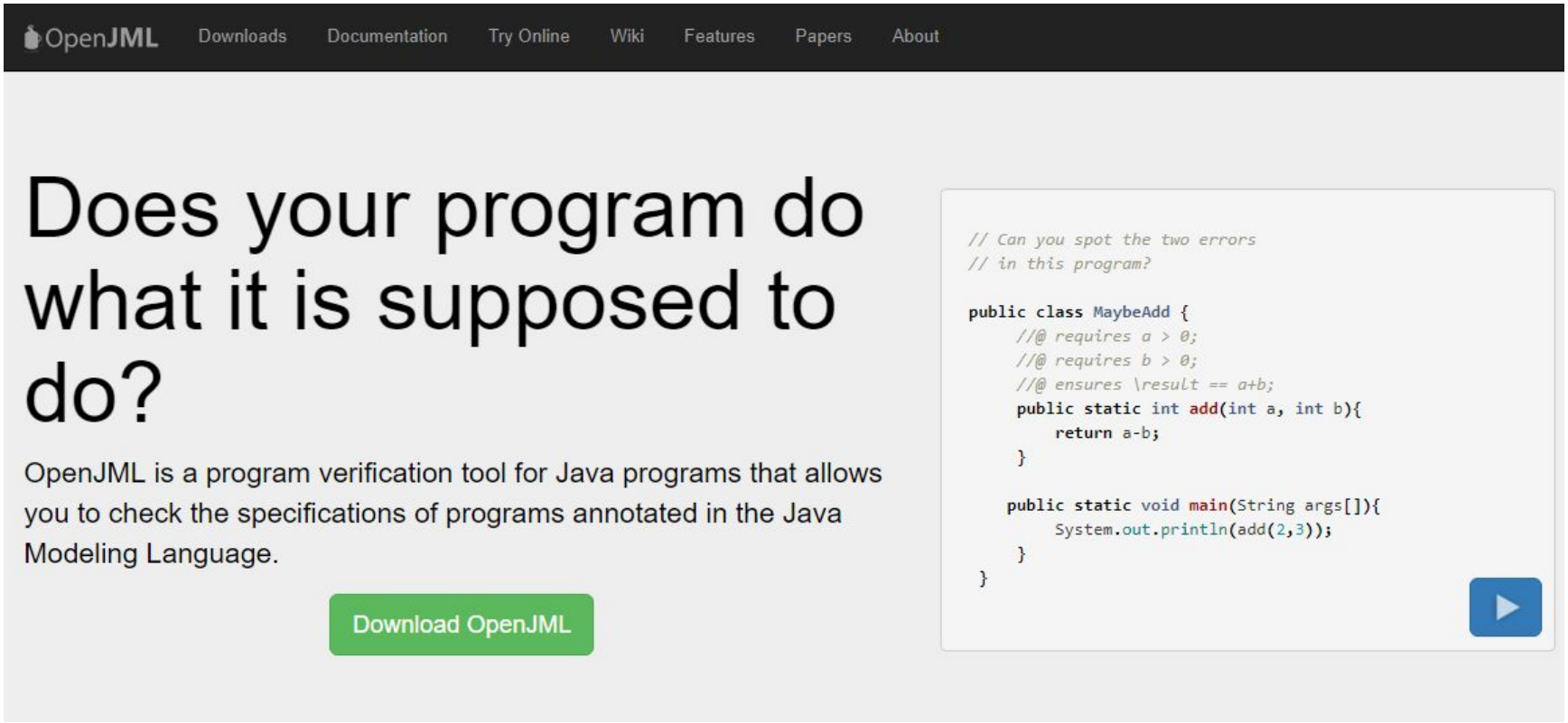
```
/*@ requires a != null
  @        && (\forall int i;
  @                0 < i && i < a.length;
  @                a[i-1] <= a[i]);
  @*/
int binarySearch(int[] a, int x) {
  // ...
}
```

# Modelos de Software

## Java Modeling Language (JML)

```
//@ refine "Person.java";

public class Person {
  private /*@ spec_public non_null @*/
    String name;
  private /*@ spec_public @*/
    int weight;

  /*@ public invariant !name.equals("")
    @          && weight >= 0; @*/

  //@ also
  //@ ensures \result != null;
  public String toString();

  //@ also
  //@ ensures \result == weight;
  public /*@ pure @*/ int getWeight();

  /*@ also
    @ requires kgs >= 0;
    @ requires weight + kgs >= 0;
    @ ensures weight == \old(weight + kgs);
    @*/
  public void addKgs(int kgs);

  /*@ also
    @ requires n != null && !n.equals("");
    @ ensures n.equals(name)
    @   && weight == 0; @*/
  public Person(String n);
}
```

# Modelos de Software



# Modelos de Software


## OpenJML (JML)

### Eclipse plug-in for OpenJML

This page provides installation instructions and a brief overview of the functionality of the OpenJML Eclipse plugin. More detailed information about the plugin is available in the Help documentation that is installed with the plugin.

### Installing

OpenJML provides a standard Eclipse plug-in update site at http://jmlspecs.sourceforge.net/openjml-updatesite. The instructions for installation are given here.

[TBD - Add a description of installation using the dropins folder]

### Features

[TBD - this section needs review and updating]

#### *MENUBAR AND TOOLBAR ITEMS*

The Menubar and toolbar contain these additional items:

- The `JML` menubar entry with its submenu items.
- A toolbar item marked by the JML logo (a yellow coffee cup), which initiates type-checking of a Java or JML file.
- A toolbar item marked `ESC`, which initiates static checking of selected files, classes or methods.
- A toolbar item marked `RAC`, which initiates compilation for runtime assertion checking