

# *Beetlz*

## *Software Model Consistency Checker for Eclipse*

Eva Darulová

Supervisor: Dr. Joseph Kiniry

University College Dublin

14th May 2009



# *Software Modelling*

- abstract view of software
  - implementation language independent
  - different forms:
    - high-level or very detailed
    - graphical or textual
    - formal or informal
  - examples: UML, Z, BON, JML
- precise specification of the system
- better understanding



# BON

```
effective class KEEPER
indexing
  about: "Looks after animals."
inherit
  PERSONNEL
feature
  name: STRING
  redefined getID: VALUE
  feedAnimal: BOOLEAN
    -> an: ANIMAL
  ensure
    delta animalsToLookAfter;
    an.hungry = false;
  end
feature{PERSONNEL}
  animalsToLookAfter: SET[ANIMAL]
invariant
  animalsToLookAfter.count < 50;
end
```

- a.k.a. Business Object Notation
- design process & notation
- graphical & textual
- object-oriented
- seamless development
- reversibility
- Design by Contract

→ Simple and complete



# JML

```
public class Keeper implements Personnel
  private /*@spec_public@*/ Set<Animal> animalsToLookAfter;
  private /*@spec_public@*/ String name;

  //@ invariant animalsToLookAfter.size() < 50;
  //@ constraint name == \old(name);

  @Override
  public /*@pure@*/ int getID() { ... }

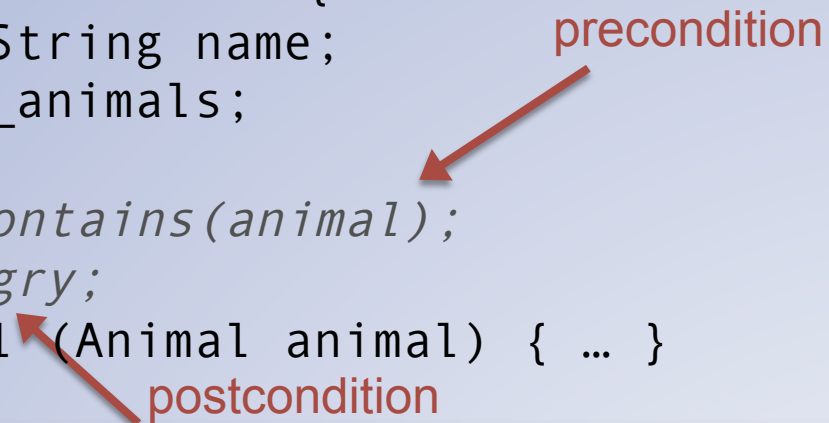
  //@ assignable animalsToLookAfter;
  //@ ensures an.hungry == false;
  public boolean feedAnimal(Animal an) { ... }
}
```

- a.k.a. Java Modelling Language
- tailored for Java
- tool support available, e.g. ESC/Java 2, jmlspecs
- supports Design by Contract



# Design by Contract

```
public class Keeper extends Personnel {  
    public /*@ non_null @*/ String name;  
    public Set < Animal > my_animals;  
  
    //@requires my_animals.contains(animal);  
    //@ensures !animal.isHungry;  
    public boolean feedAnimal (Animal animal) { ... }  
}
```



- Formal specification of software: specifies what exactly the product is supposed to do
- Set up a contract between a client class and a supplier class:
  - the client has to fulfil preconditions before calling a method
  - the supplier guarantees to fulfil the postconditions
- Classes can rely on the contract
- Specification can be used for verification purposes



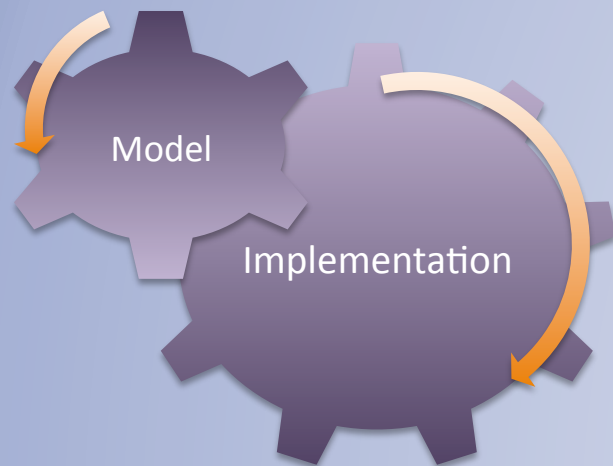
# Motivation

- usually...



- implementation changes constantly
- model needs to be updated – requires lot of effort
  - model gets forgotten once coding starts
  - benefits from modelling are lost

- better...



- consistency checks are required
- tool support is required
  - requires mapping or relation.

/ \* @



# Class Header

Deferred	abstract
Deferred with all features public and abstract	Interface
Class with distinct constant features of its own type	enum
effective	Not abstract
root	Contains public static void main(String[]) Extends Thread, implements Runnable
interfaced	All members public
reused	ignore
persistent	Implements Serializable/Externalizable
Not deferred	Static, final
ignore	strictfp

/\*@

# Class Relations

LION inherit ANIMAL	Lion extends Animal
LION inherit ANIMAL	Lion implements Animal
LION inherit ANIMAL, HUNGRY	Lion extends Animal implements Hungry
LION	Lion implements Comparable
LION inherit ANIMAL, MAMMAL	Lion implements Animal, Mammal

CAGE [ T, S, R ]	Cage < T, S, R >
CAGE [ T -> ANIMAL ] CAGE [ T -> HUNGRY ]	Cage < T extends Animal > Cage < T extends Hungry > Cage < T extends Animal & Cute >
CAGE [ ANY ]	Cage < ? >
CAGE [ ANIMAL ]	Cage < ? extends ANIMAL >
CAGE [ ANIMAL ]	Cage < ? Super ANIMAL >

/\*@



# Feature Relations

deferred	abstract
effective	not abstract
redefined	@Override annotation
not deferred	static, final, native
ignore	strictfp, synchronized, transient, volatile

public (default)	public
feature{ONE_CLASS, ANOTHER_CLASS} (restricted)	not possible
feature{MY_CLASS}	protected
feature{MY_CLUSTER}	package-private (default)
feature{NONE}	private

/\*@

# Challenges

- naming conventions
  - Java names package-unique, BON names system-unique
- implementation details
  - private methods/constructors
  - getter & setter
  - Java standard library classes
- partial translations
  - tool has to be tolerant
  - tool has to be user customisable
- impossible translations
  - static in Java
  - restricted visibility in BON



# Technical Details

Minimum Java requirements	Java 1.6
Recommended Java JVM on Mac OS X	Soylatte ( <a href="http://landonf.bikemonkey.org/static/soylatte/">http://landonf.bikemonkey.org/static/soylatte/</a> )
Target Eclipse platform	Eclipse 3.4.2
Supported interface languages	English(default), German
Number of packages	9
Number of classes	67
Total lines of code	~ 9300
Method lines of code	~ 6600



# *Implementation*

- command-line
- Eclipse plugin
  - popular programming platform
  - integrates different functionalities
  - quick and convenient use of tool
  - results presented in for user familiar fashion
- Skeleton code generation
  - print to one or individual files
  - may have to be reviewed due to not ideal mappings
- Internationalisation
  - all output strings and menu items
  - picks up language from JVM
  - supports English and German



## *So what?*

Software modelling is an effective way of developing software.

A good modelling language is needed.

Tool support is needed.

This project:

- combines these
- extends current list of consistency checkers





Java - Zoo\_with\_faults/src/zoo/zoo\_formal.bon - Eclipse SDK

File Edit Navigate Search Project Run Window Help

zoo\_formal.bon 11 Animal.java

```

cluster ENCLOSURE_CLUSTER
components
class ENCLOSURE
indexing
about: "some explanation"
feature
my_animal_capacity: VALUE
accomodateAnimal: BOOLEAN
-> animal: SEQUENCE [ANIMAL]
removeAnimal: ANIMAL
-> animal: ANIMAL

Multiple markers at this line
- moveAnimal@ENCLOSURE is missing parameter type(s) CAGE.
- moveAnimal@ENCLOSURE has unexpected arguments TERRARIUM

addAnimals
-> animals: TABLE [ANIMAL, STRING]
nameAnimal
-> name: STRING
end

```

13 errors, 5 warnings, 0 others

Description

- The mapping for animalCage
- wakeUp@SNAKE should not
- moveAnimal@ENCLOSURE
- moveAnimal@ENCLOSURE
- name@PERSONNEL expects
- PERSONNEL has redundant
- PERSONNEL is missing feat
- removeAnimal@ENCLOSUR
- removeAnimal@ENCLOSUR
- removeAnimal@ENCLOSUR
- TERRARIUM expected name
- animals@KEEPER return val
- feed@ZEBRA is missing a fu
- feed@ZEBRA is missing a fu
- feed@ZEBRA is missing poo
- BON and Java have differ
- BON and Java have differ
- Technically history contrain

zoo\_formal.bon:25: removeAnimal@ENCLOSURE is missing parameter

-> Java Warnings:

zoo\_formal.bon:18: The mapping for animalCount is not correct.

zoo\_formal.bon:228: wakeUp@SNAKE should not be redefined.

-> JML Errors:

zoo\_formal.bon:258: animals@KEEPER return value expected non\_no

zoo\_formal.bon:134: feed@ZEBRA is missing a frame condition: no

zoo\_formal.bon:134: feed@ZEBRA is missing a frame condition: al

zoo\_formal.bon:134: feed@ZEBRA is missing postcondition:

sleeping -> isFished = true

-> Jml Warnings:

3 Java errors, 2 Java warnings, 4 JML errors and 0 JML warnings

85 items selected: 13 errors, 72 warnings, 0 others

@



# Example

```
class KEEPER
```



```
public class Keeper
```

```
class KEEPER  
inherit PERSONNEL
```



```
public class Keeper  
    implements ZooPersonnel
```

```
name: STRING
```



```
/*@non_null@*/ String name;
```

```
Require  
    0 < weight;
```



```
//@requires weight > 0;
```

```
Result = a + b * c;
```



```
\result = (a + b) * c;
```

