Objects and Design in Smalltalk

5. Type Checking

Éric Tanter
etanter@dcc.uchile.cl





Type Checking

- try to detect "type errors"
  - an object is being used at a place only appropriate to objects of other types or classes
- in OO: sending a message to an object that does not understand it
  - Smalltalk: walkback at runtime
  - C++/Java: compiler

```
Whale new + 2.7
```


Type Checking Rules

- In Smalltalk, a variable points to an object, whatever its type

```
R := Rectangle new.
T := Triangle new.
R := T.
```

- In C++/Java, variables have declared types

```
Rectangle r = new Rectangle();
Triangle t = new Triangle();
r = t;
```

On Early Type Checking

- Detecting type mismatches early
  - good for safety
  - has a price (rigidity)
- Untyped variables
  - make programs easier/quicker to modify
  - more flexible
- Endless debate
  - C++/Java and Smalltalk are two opposite poles
    - both with good reasons
  - safety is important, and so too is flexibility

ETC and Polymorphism

- Early type-checking allows variable of a given type to point to objects of subtypes
  - propagation of expectations
  - a variable cannot point to an object of a supertype
- Compromise between flexibility and rigid safety

Kinds of Polymorphism

- postal mail: either bulk mail or package
  - polymorphic design for a handleMail method
    - BulkMail>>handleMail -> discard
    - Package>>handleMail -> open
  - what happens if you find a tomato in the mailbox?
    - is there a polymorphic solution?
    - in Smalltalk, yes
    - in C++/Java, no

Kinds of Polymorphism

- need for RunTime Type Identification (RTTI)
- C++/Java only support inclusion polymorphism
  - unless Tomatoe and Mail share an artificial supertype
  - use of RTTI in C++/Java (casts, instanceof)
- Smalltalk also supports implicit polymorphism
  - just write a Tomato>>handleMail
  - polymorphism that works for types of objects that may not share supertypes
  - across type hierarchies instead of within type hierarchies

RTTI

- When implicit polymorphism is needed in Java/C++, you need RTTI
  - use them sparingly!
  - can introduce hard-to-maintain conditionals
  - antithetical to the OO style of programming

```
mailbox do: [ :m |
  m class = BulkMail
    ifTrue: [ m discard ].
  m class = Package
    ifTrue: [ m open ].
  m class = Tomato
    ifTrue: [ m eat ]]
```

fix this!