Using Genericity in the Design of a Collection Class

EECS3311 Software Design
Jackie Wang

Summer 2015
Lassonde School of Engineering, York University

Abstract

This document supplements a tutorial video that demonstrates the design of a *BOOK* class, implemented via both a bad design (a book that stores any records) and a good design (a book whose type of records is parameterized). You are expected to follow this tutorial to reproduce the project and illustrations, to complete the contracts and implementations of both designs, and to write more test cases to be confident that your software is correct.

Contents

1 A Book of Mixed Record Types

From the supplier's side (i.e., *BOOK*):

- All features use the *ANY* class for the type of records, e.g.,

  add(name: STRING; record: ANY)
  get(name: STRING): ANY

From the client's side (e.g., *TEST_BOOK*):

- Declaration of a *BOOK* object requires no commitment to the record type, e.g.,

```
book: BOOK
```

- It is completely flexible as to store new records in the book (e.g., via *add*): a single book can contain, e.g., birthdays, phone numbers, addresses, *etc.*
- However, manipulating records retrieved from the book requires explicit cast (why?): e.g.,

```
if attached DATE book.get("Jim") as jim_birthday then
  jim_birthday.make_today
end
```

2 A Book of a Consistent Record Type

From the supplier's side (i.e., *BOOK[G]*, which declares a type variable G[1]):

- All features use the declared type variable G for the type of records, e.g.,

  add(name: STRING; record: G)

  get(name: STRING): G

From the client's side (e.g., *TEST_BOOK*):

- Declaration of a *GENERIC_BOOK* object requires commitment to the record type, e.g.,

```
birthday_book: GENERIC_BOOK[DATE]
```

- It is then restricted as to what new records can be stored in the book (e.g., via *add*): a single book can contain only records whose types are consistent to the one that the client has committed at the point of declaration. For example, only *DATE* records may be added to *birthday_book*.
- Due to the restriction imposed on storage operations, manipulating records retrieved from the book does not require explicit cast (why?): e.g.,

```
birthday_book.get("Jim").make_today
```

Questions: What compile time and runtime errors, related to the type(s) of records, might occur in the two designs?

[1]The name of this type variable can be any other name that has not been used as a class name.