Lecture 13

Wednesday Oct. 25

```
class BOOK [G]
  names: ARRAY[STRING]
  records: ARRAY[ANY]
  -- Create an empty book
  make do ... end
  -- Add a name-record pair to the book
  add (name: STRING; record: ANY) do ... end
  -- Return the record associated with a given name
  get (name: STRING): ANY do ... end
end
```

DATE [G] → parameter

generic class

[G] DATE

[G] DATE

[G] DATE

avg(x, y) = 2(x + y)/2 3

parameters

Supplier.

b: BOOK[DATE] [3] create {BOOK[DATE]} b.make

```
1  birthday: DATE; phone_number: STRING
2  b: BOOK; is_wednesday: BOOLEAN
3  create {BOOK} b.make
4  phone_number := "416-677-1010"
5  b.add ("SuYeon", phone_number)
6  create {DATE} birthday.make(1975, 4, 10)
7  b.add ("Yuna", birthday)
8  is_wednesday := b.get("Yuna").get_day() = 4
```

class of DATE

STRING not a descendant

client

things returned from Book are DATE's only.

```
class BOOK [G] ANY
  names: ARRAY[STRING]
  records: ARRAY[ANY] G ANY
  -- Create an empty book
  make do ... end
  -- Add a name-record pair to the book
  add (name: STRING; record: ANY) do ... end
  -- Return the record associated with a given name
  get (name: STRING): ANY do ... end
end
```

b: Book[ANY]

b.get(..).get_day

ANY

Fix 1: make MY_Book generic

```
class MY_Book[E, F]
    imp: DICTIONARY[V, K]
    :
    :
end
```

1. V and K are not known classes
2. V and K are not parameters of the current class

Fix 2: make MY_Book not generic

```
class MY_Book
    imp: DICTIONARY[RECORD, STRING]
```

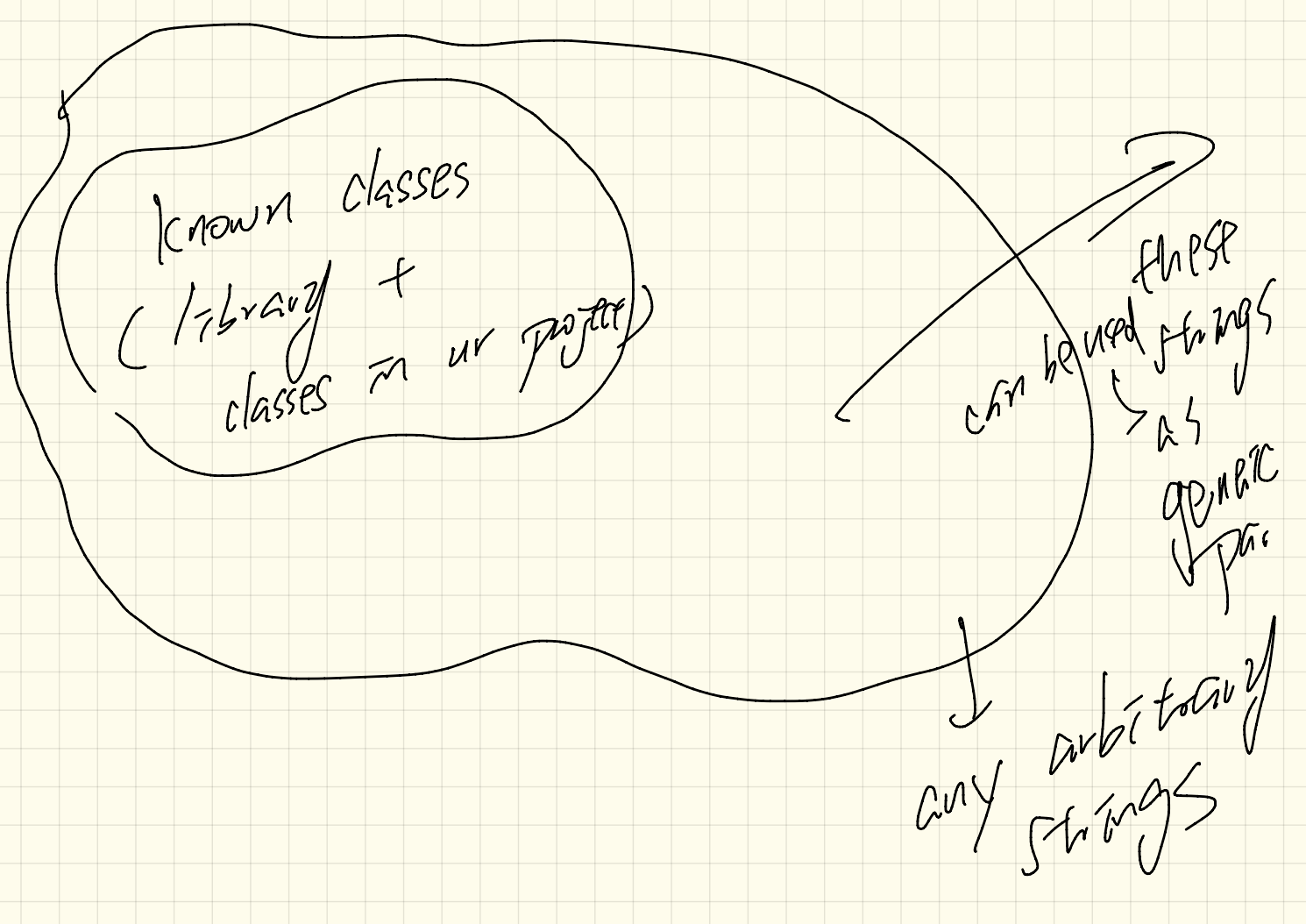
class DICTIONARY[E, F]

imp1: DIC[E, F]

imp2: DIC[E, STRING]

imp3: DIC[STRING, F]

imp4: DIC[INT, STRING]


known classes
(library +
classes in ur project)
can be used
these strings
as
generic
types
any arbitrary
strings

class Book [I_LOVE_EIFFEL, EIFFEL_IS_GREAT]

```
class SMART_PHONE
  get_reminders: LIST[EVENT]
    require
      α: battery_level ≥ 0.1 -- 10%
    ensure
      β: ∀e : Result | e happens today
end
```

T F
α ✓ γ2

10%

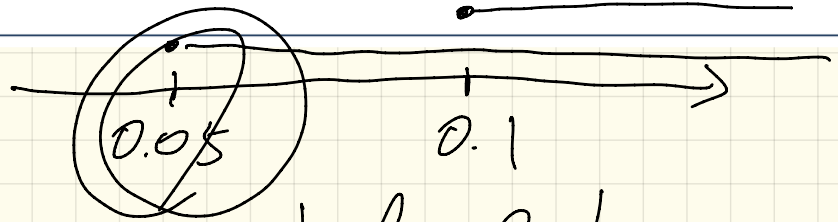
```
class IPHONE_6S_PLUS
inherit SMART_PHONE redefine get_reminders end
  get_reminders: LIST[EVENT]
    require else
      γ: battery_level ≥ 0.05 -- 5%
    ensure then
      δ: ∀e : Result | e happens today between 9am and 5pm
end
```

bad design: γ2 : b_l ≥ 0.15

if this is good!

① α ⇒ γ

② γ ⇒ α ✗ e.g. b_l = 0.06



50% 50%

1. Reduce % on programming

2. Reduce % on written