

Design by contract

- Object-Oriented Software Construction by Bertrand Meyer, Prentice Hall
- The presence of a precondition or postcondition in a routine is viewed as a contract.

4/21/98

Design by Contract

1

Rights and obligations

- Parties in the contract: class and clients
- require pre, ensure post with method r: **If you promise to call r with pre satisfied then I, in return, promise to deliver a final state in which post is satisfied.**
- Contract: entails benefits and obligations for both parties

4/21/98

Design by Contract

2

Rights and obligations

- Precondition binds clients
- Postcondition binds class

4/21/98

Design by Contract

3

Example

Contract for <i>push</i> of class Stack	Obligations	Benefits
Client Programmer	Only call <i>push(x)</i> on a non-full stack	Get <i>x</i> added as new stack top on return (<i>top</i> yields <i>x</i> , <i>nb_elements</i> increased by 1)
Class Implementor	Make sure that <i>x</i> is put on top of stack	No need to treat cases in which the stack is already full

4/21/98

Design by Contract

4

If precondition is not satisfied

- If client's part of the contract is not fulfilled, class can do what it pleases: return any value, loop indefinitely, terminate in some wild way.
- Advantage of convention: simplifies significantly the programming style.

4/21/98

Design by Contract

5

Source of complexity

- Does data passed to a method satisfy requirement for correct processing?
- Problem: no checking at all or: multiple checking.
- Multiple checking: conceptual pollution: redundancy; complicates maintenance
- Recommended approach: use preconditions

4/21/98

Design by Contract

6

Class invariants and class correctness

- Preconditions and postconditions describe properties of individual routines
- Need for global properties of instances which must be preserved by all routines
- $0 \leq \text{nb_elements}; \text{nb_elements} \leq \text{max_size}$
- `empty=(nb_elements=0);`

4/21/98

Design by Contract

7

Class invariants and class correctness

- A class invariant is an assertion appearing in the invariant clause of the class.
- Must be satisfied by all instances of the class at all “stable” times (instance in stable state):
 - on instance creation
 - before and after every remote call to a routine (may be violated during call)

4/21/98

Design by Contract

8

Class invariants and class correctness

- A class invariant only applies to public methods; private methods are not required to maintain the invariant.

4/21/98

Design by Contract

9

Invariant Rule

- An assertion I is a correct class invariant for a class C iff the following two conditions hold:
 - The constructor of C , when applied to arguments satisfying the constructor's precondition in a state where the attributes have their default values, yields a state satisfying I .
 - Every public method of the class, when applied to arguments and a state satisfying both I and the method's precondition, yields a state satisfying I .

4/21/98

Design by Contract

10

Invariant Rule

- Precondition of a method may involve the initial state and the arguments
- Postcondition of a method may only involve the final state, the initial state (through old) and in the case of a function, the returned value.
- The class invariant may only involve the state

4/21/98

Design by Contract

11

Invariant Rule

- The class invariant is implicitly added (anded) to both the precondition and postcondition of every exported routine
- Could do, in principle, without class invariants. But they give valuable information.
- Class invariant acts as control on evolution of class
- A class invariant applies to all contracts between a method of the class and a client

4/21/98

Design by Contract

12

Definitions

- Class C
- INV class invariant
- method r : $pre_r(x_r)$ precondition; $post_r$ postcondition
- x_r : possible arguments of r
- B_r : body of method r
- $Default_C$: attributes have default values

4/21/98

Design by Contract

13

Correctness of a class

- A class C is said to be correct with respect to its assertions if and only if
 - For every public method r other than the constructor and any set of valid arguments x_r :
 $\{INV \text{ and } pre_r(x_r)\} B_r \{INV \text{ and } post_r\}$
 - For any valid set of arguments x_C to the constructor:
 $\{Default_C \text{ and } pre_C(x_C)\} B_C \{INV\}$

4/21/98

Design by Contract

14

How to prove correctness

- A complex story

Verifiable Programming

- Reason about imperative sequential programs such as Java programs
- Imperative program
 - defines state space
 - defined by collection of typed program variables
 - are coordinate axis of state space
 - pattern of actions operating in state space

4/21/98

Design by Contract

15

The End

4/21/98

Design by Contract

16