Programming by Contract

Preconditions
Postconditions

(See Bertrand Meyer's classic book: "Object-oriented Software Construction", Prentice-Hall 1988/1997).

[original slides by M.Utting]

Motivation

- Want precise specifications of classes.
  - Class will be clearly documented.
  - Makes implementers job easier.
  - Callers (clients) know precisely how/when to call each method and what it will do.
  - Computer can check correctness of class and its clients. Automatic bug catching!

Contracts

- An agreement between two parties:
  - The ***provider*** provides a service/product.
  - The ***client*** uses that service/product.
    - **(and usually pays for it!).**
- Each side has obligations (and benefits):
  - *Precondition* = client's obligations.
  - *Postcondition* = provider's obligations.

Alternative definitions

- A precondition describes the properties that must hold whenever a procedure is called.
- A postcondition describes the properties that the procedure guarantees when it returns.

Contract: *If you promise to call me with Pre satisfied, then I promise to return with Post satisfied.*

Sort example

- static void Arrays.sort(int[] intArray)
  - Precondition: intArray is a properly initialized array of ints
  - Postcondition: the array will be sorted in ascending order in-place (i.e. the array will be modified!)

More Example Contracts (?)

- Frame: void paint(Graphics g);
- Stack: public Object pop();
- Stack: public Object push(Object val);

Obligations and Benefits

	Obligations	Benefits
Client:	Only call Pop on non-empty stack.	Value returned was on top of the stack. (will be removed)
Stack:	Must return top of stack and shrink stack	No need to handle cases where stack is empty.

What if precondition is false?

- In a *correct* system, procedures are only called when their preconditions are true.
- If a procedure is called when its precondition is false, it may do anything!
  (The contract is broken!)
- While debugging, we can use assertions to check some preconditions.

Checking Preconditions

```
public Object pop()
{
   // check precondition.
   assert(size() > 0);

   // pop top of stack
   . . .
}
```

- This catches *client* errors.

Checking Postconditions

```
public Object pop()
{
   // pop top of stack
   int oldSize = size();
   . . .
   // check postconditions (at end)
   assert( size() == oldSize-1 );
}
```

- This catches errors *within Pop.*

Making Contracts Better

- To improve (*refine*) a contract, we can:
  - Weaken the precondition
    (require less from the client), or
  - Strengthen the postcondition
    (do more for the client).
- Note: we take the clients viewpoint!
- If contract A is refined by contract B, clients of A will be happy to use implementations of B.

Summary

- Use English pre/postconditions to document your procedures.
  - Precondition = client's responsibility
  - Postcondition = procedure's responsibility
- Some pre/postconditions can easily be written as Java predicates:
  - (in Java 1.4 and later) use assert( ) to check them.
  - Especially useful for preconditions
    (to catch errors in client code).