# Abstraction / Information Hiding / Pre & Post Conditions

Recall…

- Method = Function = Procedure

- A method is an agent you hire to do a job for you:

  - You give the agent some information (inputs)

  - Agent goes somewhere, does whatever calculations he needs

  - Agent comes back to you with the result (output)

Method Abstraction

You can think of a method as a black box that contains the detailed implementation for a specific task.

Optional arguments
for Input

Optional return
value

Method Header

Method body

Black Box
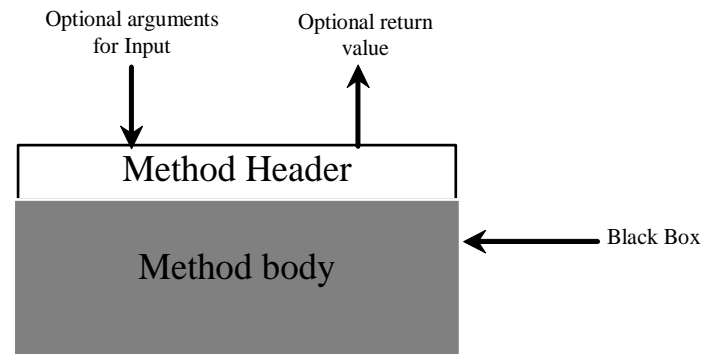
Notes:

- We *don't know* (and we *don't care* about) what calculations are done by the agent
- We are only interested in:
  - Information that is initially given to the agent (inputs)
  - Result that the agent will give back to us (output)

Information Hiding I

- The idea of an API is to expose the interface that people will use without exposing the implementation details.
- Example: java.lang.Math class

static double	log(double a)	Returns the natural logarithm (base *e*) of a double value.
static double	log10(double a)	Returns the base 10 logarithm of a double value.

- This tells you what methods to call to find the natural or base 10 logarithms of a number.
- It doesn't (and shouldn't) tell how you they do it, although it's possible to look at the code and see it.

Method pre- and post-conditions

- Method *preconditions* tell a user what the method *assumes to be true when called*, in order for the method to work correctly.

- Method *postconditions* tell
  - What *values (if any) are returned*
  - What *will be true* when the method *finishes execution*

- Method pre- and post-conditions *should be documented for any nontrivial methods*.

Method pre- and post-conditions II

- Let's look at the log method from the Java API.

log

```
public static double log(double a)
```

Returns the natural logarithm (base *e*) of a **double** value. Special cases:
- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is negative infinity.
- If the argument is **1.0**, then the result is positive zero.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

Parameters:

a - a value

Returns:

the value ln **a**, the natural logarithm of **a**.

Frequently a programmer must communicate precisely **what** a method accomplishes, without any indication of **how** the method does its work.

Can you think of a situation where this would occur ?

Example

You are the head of a programming team and you want one of your programmers to write a method for part of a project.

Example:

```
//  Precondition:  x  >=  0.
//  Postcondition:  The square root of x has
//  been written to the standard output.
public void writeSqrt( double x)



    . . .
```

Example:

```
//  Precondition:  x  >=  0.
//  Postcondition:  The square root of x has
//  been written to the standard output.
public void writeSqrt( double x)
```

- The precondition and postcondition appear as comments in your program.

- They are usually placed before the method's implementation.

Example:

```
//  Precondition:  x  >=  0.
//  Postcondition:  The square root of x has
//  been written to the standard output.
public void writeSqrt( double x)
```

Here the precondition is that

   x >= 0

whenever the method is called

Example

Which of these method calls
meet the precondition ?

🤔

```
writeSqrt( -10 );
writeSqrt( 0 );
writeSqrt( 5.6 );
```

Example

Which of these method calls
meet the precondition ?

writeSqrt(-10);
writeSqrt(0);
writeSqrt(5.6);

The second and third calls are fine, since
the argument is greater than or equal to zero.

Example

Which of these method calls
meet the precondition ?

writeSqrt(-10);
writeSqrt(0);
writeSqrt(5.6);

But this call violates the precondition
because the argument is less than 0

Example:

<pre>
// <strong style="color:red">Precondition: x >= 0.
// <strong style="color:red">Postcondition: The square root of x has
// <strong style="color:red">been written to the standard output.
public void writeSqrt(double x)
</pre>

The post-condition indicates what work the method has accomplished.

In this example the square root of x has be written after the method completed.

Another Example

// Precondition: letter is an uppercase or
// lowercase letter (in the range 'A' ... 'Z' or 'a' ... 'z') .
// Postcondition: The value returned by the
// method is true if letter is a vowel;
// otherwise the value returned by the method is
// false.
public boolean isVowel(char letter)

. . .

Another Example

//   Precondition: letter is an uppercase or
//   lowercase letter (in the range 'A' ... 'Z' or 'a' ... 'z') .
//   Postcondition: The value returned by the
//   method is true if letter is a vowel;
//   otherwise the value returned by the method is
//   false.
public boolean isVowel(char letter)

■ ■ ■

Another Example

What values will be returned
by these method calls ?

```
isVowel( 'A' );
isVowel(' Z' );
isVowel( '?' );
```

Another Example

What values will be returned by these method calls ?

```
isVowel( 'A' );
isVowel(' Z' );
isVowel( '?' );
```

true

false

Nobody knows, because the precondition has been violated.

Another Example

**What values will be returned
by these method calls ?**

```
isVowel( '?' );
```

**Violating the precondition
might even crash the program.**

Always make sure the precondition is valid . . .

- The programmer who calls the method is responsible for **ensuring that the precondition is valid** when the method is called.

AT THIS POINT, MY PROGRAM CALLS YOUR METHOD, AND I MAKE SURE THAT THE PRECONDITION IS VALID.

. . . so the post-condition becomes true at the method's end.

- The programmer who writes the method counts on the precondition being valid, and **ensures that the postcondition becomes true** at the method's end.

THEN MY METHOD WILL EXECUTE, AND WHEN IT IS DONE, THE POSTCONDITION WILL BE TRUE.
I GUARANTEE IT.

A Quiz

Suppose that you call a method, and you neglect to make sure that the precondition is valid.
Who is responsible if this inadvertently causes a 40-day flood or other disaster?

☐ You

☐ The programmer who wrote that torrential method

☐ Noah

A Quiz

Suppose that you call a method, and you neglect to make sure that the precondition is valid.
Who is responsible if this inadvertently causes a 40-day flood or other disaster?

☐ **You**

☐ The programmer who wrote that torrential method

☐ Noah

On the other hand, careful programmers also follow these rules:

- When you write a method, you should make every effort to detect when a precondition has been violated.

- If you detect that a precondition has been violated, then print an error message and halt the program…

On the other hand, careful programmers also follow these rules:

- When you write a method, you should make every effort to detect when a precondition has been violated.

- If you detect that a precondition has been violated, then print an error message and halt the program…

- …rather than causing a disaster.
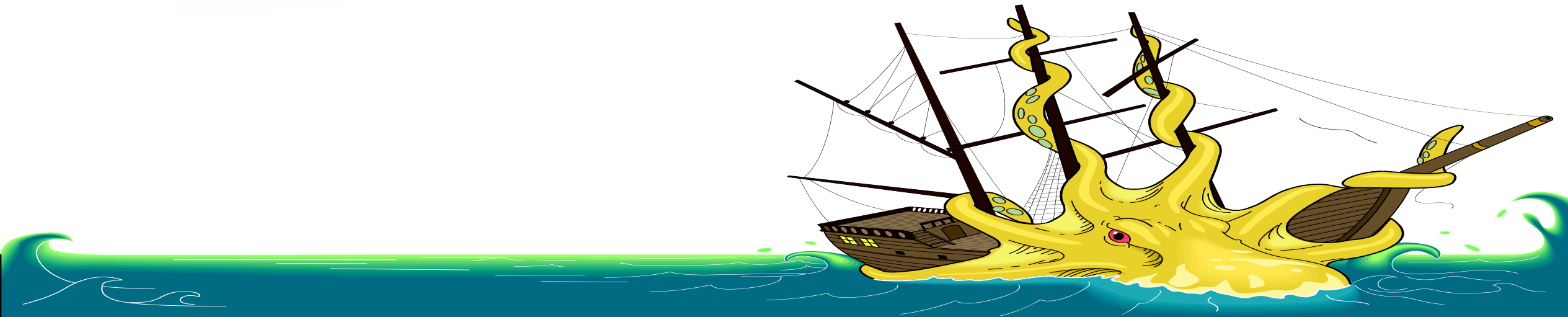
Our Previous Example

```
//   Precondition:  x  >=  0.
//   Postcondition:  The square root of x has
//   been written to the standard output.
public void writeSqrt( double x)
{
    if (x < 0)
        throw new IllegalArgumentException("Negative x");

    . . .
```

Throwing an exception can be useful

Advantages of Using Preconditions and Postconditions

- Succinctly describes the behavior of a method...

- ... without cluttering up your thinking with details of how the method works.

- At a later point, you may reimplement the method in a new way ...

- ... but programs (which only depend on the precondition/postcondition) will still work with no changes.

Summary about pre/post-conditions

Precondition

- The programmer who calls a method ensures that the precondition is valid.
- The programmer who writes a method can bank on the precondition being true when the method begins execution.

Postcondition

- The programmer who writes a method ensures that the post-condition is true when the method finishes executing.

Q3

- Consider the methods you wrote in Q2:

  – Can you write the preconditions

  – Can you write the post-conditions?

  – Can you write some statements that detect when the precondition is violated and print a message?