Practice 12. Readability

Byoung-Tak Zhang
TA: Hanock Kwak
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

http://bi.snu.ac.kr

Special Comment Blocks

- Comments written in the special format can be used in the document generating programs like *doxygen*.
- They also can be shown in the bubble popups in some IDE.

```
/**
* a normal member taking two arguments and returning an integer value.
* @param a an integer argument.
* @param s a constant character pointer.
* @see testMeToo()
* @see publicVar()
* @return The test results
*/
int testMe(int a,const char *s);
```

Xerces-C++ 3.1.2

Base64 Class Reference

Public Types

enum **Conformance** { **Conf_RFC2045**, **Conf_Schema** }

Static Public Member Functions

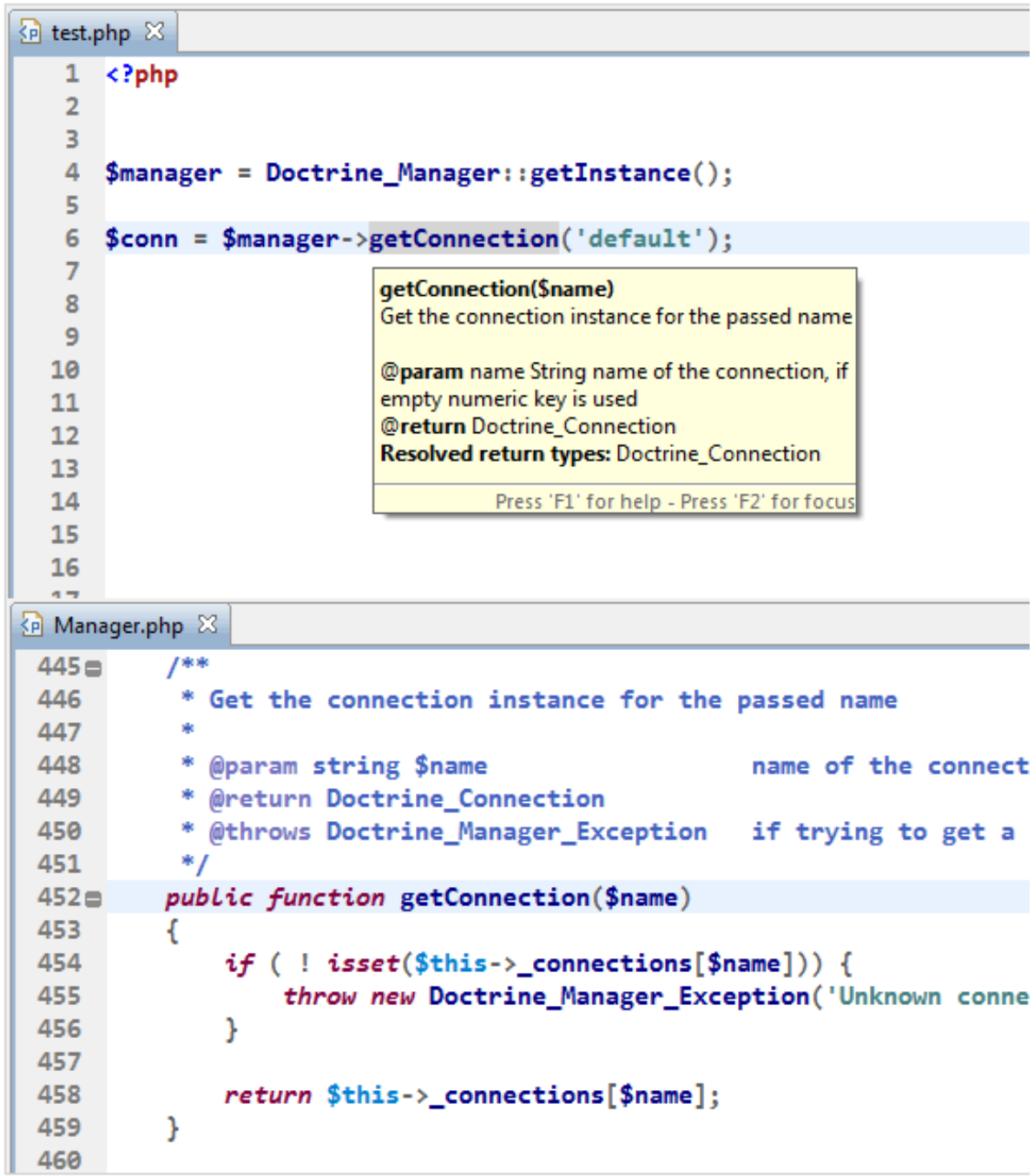
static **XMLCh** * **getCanonicalRepresentation** (const **XMLCh** *const inputData, **MemoryManager** *const memMgr=0, **Conformance** conform=**Conf_RFC2045**)
get canonical representation More...

static **XMLByte** * **encode** (const **XMLByte** *const inputData, const **XMLSize_t** inputLength, **XMLSize_t** *outputLength, **MemoryManager** *const memMgr=0)
Encodes octets into **Base64** data. More...

static **XMLByte** * **decode** (const **XMLByte** *const inputData, **XMLSize_t** *decodedLength, **MemoryManager** *const memMgr=0, **Conformance** conform=**Conf_RFC2045**)
Decodes **Base64** data into octets. More...

static **XMLByte** * **decodeToXMLByte** (const **XMLCh** *const inputData, **XMLSize_t** *decodedLength, **MemoryManager** *const memMgr=0, **Conformance** conform=**Conf_RFC2045**)
Decodes **Base64** data into octets. More...

static int **getDataLength** (const **XMLCh** *const inputData, **MemoryManager** *const memMgr=0, **Conformance** conform=**Conf_RFC2045**)
Get data length. More...


test.php
1 <?php
2
3
4 $manager = Doctrine_Manager::getInstance();
5
6 $conn = $manager->getConnection('default');
7
8
9
10
11
12
13
14
15
16
getConnection($name)
Get the connection instance for the passed name
@param name String name of the connection, if empty numeric key is used
@return Doctrine_Connection
Resolved return types: Doctrine_Connection
Press 'F1' for help - Press 'F2' for focus
Manager.php
445 /**
446 * Get the connection instance for the passed name
447 *
448 * @param string $name name of the connect
449 * @return Doctrine_Connection
450 * @throws Doctrine_Manager_Exception if trying to get a
451 */
452 public function getConnection($name)
453 {
454 if (! isset($this->_connections[$name])) {
455 throw new Doctrine_Manager_Exception('Unknown conne
456 }
457
458 return $this->_connections[$name];
459 }
460

Naming Conventions

- Naming conventions are important when your codes are shared with some other programmers.
- Programming languages and developer communities have their own traditional naming conventions.
- See
  - https://developer.gnome.org/programming-guidelines/stable/c-coding-style.html.en

Deep Nesting

- Avoid deep nesting.

```
for (i = 0; i < N; i++)
{
    for (j = 0; j < M; j++)
    {
        if (b[i][j] == 'A')
        {
            if (b[i][j+1] != 'B')
                x = 1;
            else
                x = 2;
        }
    }
}
```

```
for (i = 0; i < N; i++)
{
    for (j = 0; j < M; j++)
    {
        if (b[i][j] != 'A')
            continue;

        if (b[i][j+1] != 'B')
            x = 1;
        else
            x = 2;
    }
}
```

Code Grouping

```
#include <stdio.h>

int main()
{
    int array[100], minimum, size, c, location = 1;

    printf("Number of elements:\n");
    scanf("%d",&size);

    printf("Enter %d integers\n", size);

    for ( c = 0 ; c < size ; c++ )
        scanf("%d", &array[c]);

    minimum = array[0];

    for ( c = 1 ; c < size ; c++ )
    {
        if ( array[c] < minimum )
        {
           minimum = array[c];
           location = c+1;
        }
    }

    printf("Minimum element %d.\n", minimum);
    return 0;
}
```

Not grouped

Code Grouping

```
#include <stdio.h>

int main()
{
    int array[100], minimum, size, c,
location = 1;

    // read an array from the user
    printf("Number of elements:\n");
    scanf("%d",&size);

    printf("Enter %d integers\n", size);

    for ( c = 0 ; c < size ; c++ )
        scanf("%d", &array[c]);

    // find the minimum of the array
    minimum = array[0];

    for ( c = 1 ; c < size ; c++ )
    {
        if ( array[c] < minimum )
        {
            minimum = array[c];
            location = c+1;
        }
    }

    // print the minimum
    printf("Minimum element %d.\n", minimum);
    return 0;
}
```

Simplification

- Simplify the complicated expressions using local variables or functions.

```
if ('a' < ch && ch < 'z')
...
```

```
if ( islower(ch) )
...
```

```
k = a[x[i][j] + dx[c]][y[i][j] + dy[c]];
```

```
int nx = x[i][j] + dx[c];
int ny = y[i][j] + dy[c];
k = a[nx][ny];
```