Приложение
к приказу НИУ ВШЭ
от 28.05.2015 № 6.18.1-01/2805-15

Национальный исследовательский университет
«Высшая школа экономики»

СТАНДАРТ
разработки программного обеспечения и комментирования программных кодов информационных систем

Москва

1. Общие положения

1.1 Соблюдение единых правил оформления программного кода значительно облегчает читаемость кода и позволяет обеспечивать высокую эффективность технической поддержки информационных систем и процессов реализации доработок в информационных системах.

1.2 Настоящий «Стандарт разработки программного обеспечения и комментирования программных кодов информационных систем НИУ ВШЭ» (далее – Стандарт) определяет основные требования к оформлению программных кодов при разработке программных кодов для любых информационных систем НИУ ВШЭ.

1.3 Стандарт описывает принципы построения программного кода в части:

1.3.1 структурирования программных кодов в соответствии с правилами, изложенными в настоящем документе.

1.3.2 написания содержательных пояснений (комментариев) к программным кодам в соответствии с правилами, изложенными в настоящем документе.

1.4 Стандарт является обязательным для любых разработок новых информационных систем, а также любых доработок существующих информационных систем в рамках НИУ ВШЭ. Допускается отклонение от настоящего Стандарта по предварительному письменному разрешению Старшего Директора, курирующего направление ИТ НИУ ВШЭ.

1.5 Все разработчики, создающие программный код в интересах НИУ ВШЭ, обязаны соблюдать требования и рекомендации Стандарта.

1.6 Каждое требование и рекомендация Стандарта действительны только для тех языков программирования, где допускается их использование, и они технически выполнимы.

1.7 Непосредственные руководители разработчиков программного кода для НИУ ВШЭ, а также лица, курирующие направление разработки ИТ в интересах НИУ ВШЭ, обязаны осуществлять контроль соблюдения настоящего Стандарта.

2. Разработка программного кода и объектов баз данных

При разработке программного кода и баз данных в интересах НИУ ВШЭ должны соблюдаться следующие единые правила его оформления:

2.1 Для названий любых элементов программного кода и объектов баз данных (модулей[1], классов и их методов, функций, процедур, пакетов, триггеров, таблиц и их столбцов, переменных, констант, последовательностей (SEQUENCE), представлений (VIEW)), следует использовать содержательные имена, позволяющие сделать вывод о назначении данных элементов.

2.2 При использовании в названиях классов нескольких слов каждое слово следует начинать с заглавной буквы, например, «ServerConnector».

2.3 В названиях методов и событий класса первое слово следует начинать с маленькой буквы, остальные слова – с большой, например, «getRecordByNumber» или «onMouseDown».

2.4 Для языков C#, VB.NET: в названиях публичных методов класса первое слово следует начинать с большой буквы, например, «GetRecordByNumber», а не публичных – с маленькой, например, «getRecordByNumber».

2.5 При использовании в названиях функций, процедур, пакетов, триггеров, нескольких слов каждое слово следует начинать с заглавной буквы, например, «PrintMatrix».

2.6 Для языков JAVA, PHP: в названиях функций, первое слово следует начинать с маленькой буквы, остальные слова – с большой, например, «getRecordByNumber».

2.7 Константы следует именовать в верхнем регистре, например, «MAXCOUNT».

2.8 При использовании в названиях экземпляров класса, переменных, каждое слово следует начинать с заглавной буквы.

2.9 При использовании в названиях таблиц и их столбцов нескольких слов каждое слово рекомендуется разделять знаком подчеркивания.

2.10 Для языков JAVA, PHP: Переменные, экземпляры класса, таблицы, столбцы следует именовать в нижнем регистре. При использовании нескольких слов в названии переменных их следует разделять знаком подчеркивания.

2.11 Имена параметров (при их наличии) методов класса, функций, процедур должны иметь свой унифицированный префикс, чтобы отличаться от используемых в программном коде переменных.

2.12 Имена методов класса или функций, которые возвращают значения, должны иметь свой унифицированный префикс «get», например, “getStatus”.

2.13 Имена методов класса, функций или процедур, которые устанавливают значения, должны иметь свой унифицированный префикс «set», например, “setStatus”.

2.14 Имена классов и модулей, а также пакетов, процедур и функций, которые предназначены для формирования отчетов, должны иметь свой унифицированный префикс, например, «Report».

2.15 Для улучшения наглядности структуры программного кода следует использовать пробелы и отступы:

[1] Здесь и далее под модулем подразумевается файл, содержащий набор классов или функций.

2.15.1 Пробелы – для лучшего обозначения отдельных элементов кода в строке.

2.15.2 Горизонтальные отступы – для лучшего обозначения вложенности функциональных блоков многострочного кода. При формировании горизонтальных отступов рекомендуется использовать пробел и избегать использования символа табуляции.

2.15.3 Вертикальные отступы – для отделения логических блоков программного кода между собой.

Пример программного кода без соблюдения правил оформления кода (язык C++):

```
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <time.h>
void printMatrix(int[][10], const int);
using namespace std;
int main()
{
const int SIZE=10;
int matrix[SIZE][SIZE];
int temp;
srand(time(NULL));
for(int i=0;i<SIZE;i++)
for(int j=0;j<SIZE;j++)
matrix[i][j]=1+rand()%100;
for(int N=1;N<SIZE*SIZE;N++)
{
for(int i=0;i<SIZE;i++)
{
for(int j=0;j<SIZE-1;j++)
{
if(matrix[i][j+1]<matrix[i][j])
{
temp=matrix[i][j+1];
matrix[i][j+1]=matrix[i][j];
matrix[i][j]=temp;
}}}}
printMatrix(matrix,SIZE);
return 0;
}
void printMatrix(int mx[][10],const int SIZE)
{
for(int i=0;i<SIZE;i++)
{
cout << endl;
for(int j=0;j<SIZE;j++)
cout << setw(4) << mx[i][j];
cout << endl;
}}
```

Пример программного кода с соблюдением правил оформления кода (язык C++):

```
// SortArray.cpp Сортировка строк двумерного массива методом пузырька
// Создан 25.02.2015 Сидоров С.С. Заявка №12344

#include <iostream>
#include <iomanip>
```

```
#include <stdlib.h>
#include <time.h>

//печать массива
void printMatrix
(int[][10], // сортируемый массив
 const int  // размерность массива
);
using nameSpace std; //установка стандартного пространства имен

int main()
{
 const int SIZE = 10; //размерность массива
 int matrix[SIZE][SIZE]; //определяем массив
 int temp; //временная переменная
 srand(time(NULL));//определяем исходную точку для генератора случайных чисел

    //заполняем массив случайным образом
    for(int i = 0; i < SIZE; i++)
        for(int j = 0; j < SIZE; j++)
            matrix[i][j] = 1 + rand() % 100;

    //сортируем пузырьком
    for(int N = 1; N < SIZE * SIZE; N++)
    {
        for(int i = 0; i < SIZE; i++)
        {
            for(int j = 0; j < SIZE - 1; j++)
            {
                //если нужно скорректировать порядок сортировки
                if(matrix[i][j + 1] < matrix[i][j])
                {
                 //делаем обмен элементов массива местами
                   temp = matrix[i][j + 1];
                   matrix[i][j + 1] = matrix[i][j];
                   matrix[i][j] = temp;
                }
            }// for(int j = 0; j < SIZE - 1; j++)
        }// for(int i = 0; i < SIZE; i++)
    }// for(int N = 1; N < SIZE * SIZE; N++)
    //выводим массив на печать
    printMatrix(matrix, SIZE);

    return 0;
}// end of main SortArray.cpp

//печать массива
void printMatrix(int varMX[][10], const int SIZE)
{
   for(int i = 0; i < SIZE; i++)
   {
       cout << endl;

       for(int j = 0; j < SIZE; j++)//печать j-массива на одной строке
           cout << setw(4) << varMX[i][j];

       cout << endl; // каждый j-массив на отдельной строке
   }// for(int i = 0; i < SIZE; i++)
}// end of printMatrix
// end of SortArray.cpp
```

2.16 Разрабатываемые гипертекстовые документы (HTML) рекомендуется проверять на соответствие стандартам HTML с помощью сервиса W3C «HTML validator» http://validator.w3.org/ на отсутствие ошибок.

2.17 Разрабатываемые каскадные таблицы стилей (CSS) рекомендуется проверять на соответствие стандартам CSS с помощью сервиса W3C «CSS validator» http://jigsaw.w3.org/css-validator/ на отсутствие ошибок.

3. Комментирование программного кода и объектов баз данных

3.1 Общие правила комментирования

3.1.1 Комментирование выполняется для всех элементов программного кода и объектов базы данных (далее - БД), для которых это технически возможно (модули, классы и их методы, функции, процедуры, пакеты, триггеры, таблицы и их столбцы, переменные, константы, а также последовательности (SEQUENCE) и представления (VIEW)).

3.1.2 Все комментарии пишутся на русском языке. В комментариях не допускается использование ненормативной, оскорбительной или дискриминационной лексики, а также слов, не относящихся к техническому или деловому лексикону.

3.1.3 В комментариях не допускается наличие информации, не несущей смысловую нагрузку к пояснению сути алгоритма программного кода. Например, недопустимы комментарии вида: «это гениальный код», «таблица1», «!№;%:?*» и подобные им.

3.1.4 Если для языка программирования применяется система контроля версий, то комментарии вида: дата создания; дата модификации; автор создания; автор модификации - допускается вести в ней.

3.2 Комментирование программного кода

3.2.1 При создании модуля, класса, метода класса, функции, процедуры, пакета, триггера, представления, последовательности, обязательны следующие комментарии:

- Описание назначения.
- Дата создания.
- Автор создания.
- Номер заявки в системе учета заявок на доработку, в соответствии с которой осуществляется разработка.
- Описание входных и выходных параметров (при их наличии).
- Описание назначения вложенных элементов программного кода (при их наличии).
- Описание входных и выходных параметров вложенных элементов программного кода (при их наличии).

3.2.2 Для вложенных в класс методов и событий допускается не указывать комментарии, которые полностью повторяют общий комментарий в начале класса, например, номер заявки, дата и автор создания.

3.2.3 Для вложенных в модуль классов, процедур и функций допускается не указывать комментарии, которые полностью повторяют общий комментарий в начале модуля, например, номер заявки, дата и автор создания.

3.2.4 Для вложенных в пакет процедур, функций, допускается не указывать комментарии, которые полностью повторяют общий комментарий в начале пакета, например, номер заявки, дата и автор создания.

3.2.5 При модификации модуля, класса, функции, процедуры, пакета, триггера обязательны следующие комментарии:

- Дата модификации.
- Автор модификации.
- Номер заявки в системе учета заявок на доработку, в соответствии с которой осуществляется доработка.
- Цель модификации.
- Описание входных и выходных параметров (при их изменении).
- Описание назначения вложенных элементов программного кода (при их изменении).
- Описание входных и выходных параметров вложенных элементов программного кода (при их изменении).

3.2.6 Изменения программного кода, производимые в рамках внутренней отладки, не фиксируются.

3.2.7 Для языков C#, VB.NET: комментарии к программному коду рекомендуется оформлять в виде XML-комментариев.

3.2.8 При использовании стороннего модуля, класса, функции, процедуры или пакета, следует добавлять комментарий с описанием его назначения за исключением часто используемых стандартных модулей, классов, функций, процедур или пакетов.

Пример правильного оформления комментариев спецификации пакета БД ORACLE (язык PL/SQL):

```
CREATE OR REPLACE PACKAGE SheduleStudents AS
-- Пакет для расчета расписания и назначения по курсам
-- Создан 25.02.2015 Иванов И.И. Заявка №12345

      FUNCTION GetStudentCourses
-- Функция для получения списка расписания по студенту
-- Создана 27.02.2015 Иванов И.И. Заявка №12399
-- Модификация 01.03.2015 Петров П.П. Заявка №13567 Корректировка алгоритма
поиска учащихся
(
pStudent INT,  -- идентификатор студента
pSemester INT -- номер семестра
      )
      RETURN VARCHAR2; -- на выходе список занятий

END SheduleStudents;
```

Пример правильного оформления комментариев при подключении сторонних модулей (язык Java):

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.text.SimpleDateFormat; // класс для преобразования формата дат
```

3.2.9 Для модуля, функции, процедуры, триггера, представления или пакета обязательны следующие комментарии:

- Описание используемых переменных, массивов, классов (при их наличии).
- Метка начала модификации
- Метка окончания модификации
- Номер заявки, в соответствии с которой осуществляется доработка.
- Описание алгоритма работы в бизнес-терминах и в объеме, достаточном для понимания работы алгоритма другим разработчиком и/или через продолжительный промежуток времени.

3.2.10 Для удобства работы с большой глубиной вложенности программных структур рекомендуется добавлять следующие комментарии:

- Метка окончания каждого ветвления алгоритма с указанием типа ветвления и краткого текста с условием.
- Метка окончания каждого цикла с указанием типа цикла и названия переменной-счетчика.
- Метка окончания модуля, класса, метода класса, события класса, функции, процедуры, триггера, пакета, с указанием названия закончившегося элемента программного кода, если указание названия не предусмотрено синтаксисом используемого языка.

Пример правильного оформления комментариев тела пакета БД ORACLE (язык PL/SQL):

```
CREATE OR REPLACE PACKAGE BODY SheduleStudents AS

FUNCTION GetStudentCourses
(
 pStudent INT,
 pSemester INT
      )
      RETURN VARCHAR2 IS
begin
-- начало модификации №13567
/* -- февральская версия с ошибкой в методе подсчета студентов
   -- в случае отсутствия ошибок удалить после 01.06.2015
Select count(*)
Into k
From StudentsSchedule
Where Semester = pSemester
;
*/
-- определяем общее число учащихся студентов в семестре
Select count(distinct (StudentId))
```

```
Into k
From StudentsSchedule
Where Semester = pSemester
;
-- конец модификации №13567
...
END GetStudentCourses;

END SheduleStudents;
```

3.2.11 Комментарии к тексту программного кода должны быть лаконичными и пояснять суть и назначение программного кода, а не его техническую составляющую. Например, вместо комментария «увеличиваем счетчик на 1» нужно писать «переходим к обработке следующего факультета». Текст комментариев не должен иметь сокращений, не используемых в НИУ ВШЭ, а также понимание которых может быть неоднозначным или затруднительным.

3.2.12 На этапе промышленной эксплуатации не допускается наличие комментариев, содержащих закомментированный программный код. В случае обоснованного наличия закомментированного кода в тексте программы должен быть помещен комментарий, объясняющий необходимость наличия закомментированного кода в тексте программы. В этом случае к смысловому пояснению причины наличия закомментированного кода добавляется дата его возможного удаления, которая должна быть позднее даты изменения не менее чем на три месяца.

Пример правильного оформления комментариев фрагмента кода (язык Pascal):

```
// SortArray.pas Сортировка элементов массива
// Создан 20.10.2014 Иванов И.И. Заявка №0978

// Модификация 11.11.2014 Петров П.П. Заявка №10054
// Оптимизация алгоритма: сокращение количества используемых переменных
const
    M = 10; // размерность массива
var
    arr: array[1..M] of integer; // массив для сортировки
// начало модификации №10054
// исключаем использование k. в случае отсутствия ошибок удалить после
11.02.2015
//  i, j, k: integer; // два индекса и временная переменная
    i, j: integer; // два индекса
// конец модификации №10054
begin
  randomize;
  write ('Заполняем исходный массив случайными числами: ');
  for i := 1 to M do begin
      arr[i] := random(256);
      writeln (arr[i]:4);
  end;// for i := 1 to m
  writeln;
  //сортировка
```

```
  for i := 1 to M-1 do
      for j := 1 to M-i do
          if arr[j] > arr[j+1] then begin

// -- начало модификации №10054
{
// старая версия. в случае отсутствия ошибок удалить после 11.02.2015
              k := arr[j];
              arr[j] := arr[j+1];
              arr[j+1] := k
}
              arr[j+1] := arr[j] + arr[j+1];
              arr[j] := arr[j+1] - arr[j];
              arr[j+1] := arr[j+1] - arr[j];

//-- конец модификации №10054
          end;// if (arr[j] > arr[j+1])
              // for j := 1 to m-i
              // for i := 1 to m-1
  write ('Отсортированный массив: ');
  for i := 1 to M do
      writeln (arr[i]:4);
end. // end of SortArray.pas
```

Пример правильного оформления комментариев фрагмента кода (язык Java):

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

/**
 * Класс-коннектор для обеспечения взаимодействия с сервером
 * Подключается через {@link Socket}
 * посылает GET-запрос
 * использует {@link ObjectInputStream} и {@link ObjectOutputStream}.
 */
  public class ServerConnector
  {
    // адрес подключения
    private String ip = null;
    // номер порта
    private int port;

    /**
     * Конструктор
     *
     * @param hostname (Имя сервера)
     * @param portNumber (Порт)
     */
    public ServerConnector(String hostname, int portNumber)
    {
        this.ip = hostname;
        this.port = portNumber;
```

```
    }
    /**
     * Метод отправляет GET-запрос к серверу для получения записей БД
     * Возвращает массив записей
     *
     * @param recordNumber (id записи)
     * @return array (массив элементов)
     * @throws RecordNotFoundException
     *         Если запись не найдена, она отмечается как удаленная
     */
    public String[] getRecordByNumber(long recordNumber)
            throws RecordNotFoundException
    {
        ServerCommand cmd = new ServerCommand(CommandType.GET, recordNumber);
        // инициализация переменных
  ServerResult result = null;
        Socket socket = null;
        ObjectOutputStream oos = null;
        ObjectInputStream ois = null;
        try
  {   //Открываем подключение
            socket = new Socket(ip, port);
            oos = new ObjectOutputStream(socket.getOutputStream());
            ois = new ObjectInputStream(socket.getInputStream());

            //Шлем запрос
            oos.writeObject(command);

            //Получаем ответ
            result = (ServerResult) ois.readObject();

        } catch (Exception e)
    {
            result = new ServerResult(e);

          } finally
{
             /* Закрываем подключение
              ……
             */
            }// end of try
        if (result.getError() != null)
  {
//обработка ошибок
            if (result.getError()instanceof RecordNotFoundException)
{   // если запись не найдена, считаем ее удаленной
                throw (RecordNotFoundException) result.getError();
            }
else
{   // ошибка при работе базой данных
                throw new DBException(result.getError());
            }
        }// end of if(result.getError() != null)
        return result.getData();
    }// end of getRecordByNumber(long recordNumber)
}// end of ServerConnector
// end of модуль обмена данными через Socket
```

3.2.13 В программном коде не рекомендуется указывать идентификационные коды позиций (ID) и конкретные значения из справочников (кампусы, факультеты, группы и т.д.), если это может привести к необходимости корректировки программного кода в результате изменения пользователями значений

справочников информационной системы. Вместо этого рекомендуется выносить соответствующие настройки для таких значений на уровень пользовательского интерфейса (в конфигурационные таблицы или файлы) для возможности модификации их пользователями без привлечения служб ДИТ НИУ ВШЭ.

Пример:

В БД есть справочник вида:

ORGANISATION_ID	ORGANISATION_NAME	ORGANISATION_TYPE
2135	Нижегородский филиал НИУ ВШЭ	НИУ ВШЭ
7685	Санкт-Петербургский филиал НИУ ВШЭ	НИУ ВШЭ
9085	Пермский филиал НИУ ВШЭ	НИУ ВШЭ
…	…	…

Неправильное использование ссылок на значения справочников:

```
If organisation_id = 2135 or organisation_id = 7685 or organisation_id = 9085
then
```

Правильное использование ссылок на значения справочников:

```
If GetOrgType(organisation_id) = 'НИУ ВШЭ' then
```

3.3 Комментирование объектов БД, предназначенных для хранения данных

Комментирование таблиц и их столбцов включает в себя комментарии двух видов:

- Комментарий, описывающий назначение таблицы.
- Комментарии, описывающие назначение каждого столбца таблицы.

Пример комментирования таблицы БД MySQL и ее столбцов (язык SQL):

```
CREATE TABLE employees (
id int not null auto_increment COMMENT 'Уникальный идентификатор сотрудника',
first_name varchar(50) DEFAULT NULL COMMENT 'Имя',
last_name varchar(50) DEFAULT NULL COMMENT 'Фамилия',
PRIMARY KEY (id)
) DEFAULT CHARSET=cp1251 COMMENT='Имена сотрудников';
```

Пример комментирования таблицы БД ORACLE и ее столбцов (язык SQL):

```
CREATE TABLE employees
    ( employee_id    NUMBER,
      first_name     VARCHAR2(50),
      last_name      VARCHAR2(50),
salary          NUMBER(8)
    );
COMMENT ON TABLE employees IS 'Имена и доход сотрудников';

COMMENT ON COLUMN employees.employee_id IS 'Уникальный идентификатор (ID)
сотрудника';
```

```
COMMENT ON COLUMN employees.first_name IS 'Имя сотрудника';
COMMENT ON COLUMN employees.last_name IS 'Фамилия сотрудника';
COMMENT ON COLUMN employees.salary IS 'Зарплата сотрудника';
```