UNIVERSIDAD VERACRUZANA
FACULTAD DE ESTADISTICA E INFORMATICA

PROGRAMA EDUCATIVO

INGENIERIA DE SOFTWARE

EXPERIENCIA EDUCATIVA

PRINCIPIOS DE CONSTRUCCION

DOCENTE

JUAN XARLOS PEREZ ARRIAGA

TRABAJO

ESTADNAR DE PROYECTO

ESTUDIANTES

JOSÉ MANUEL IÑIGUEZ LÓPEZ
SETH NOE DÍAZ DÍAZ
MIDGUET ARTURO GARCIA TORRES

6 DE MARZO DE 2020

INDICE

1. Introducción

Para crear un proyecto de software no basta con realizar un sistema que funciona, sino que se debe realizar un sistema que cualquier persona pueda leer o editar con facilidad, para esto es necesario seguir ciertos estándares de programación los cuales nos servirán para desarrollar sistemas de calidad y entrenarnos para pertenecer a un equipo de trabajo.

2. Propósito

El propósito de seguir estos estándares será facilitarla lectura del código además de trabajar según el equipo de trabajo el cual seguirá el mismo estándar de programación evitando tiempos muertos en depurar código por violaciones del estándar.

3. Buenas practicas

a. Bloques de comentarios

Los comentarios se deben separar en bloques delimitados por su operador. Deben usarse para describir la funcionalidad del código.

Correcto:

```
/*
This is a comment
*/
```

Incorrecto:

```
/* This is a comment
*/
```

b. Comentarios de línea

Los comentarios de línea deben ser usados para explicar el funcionamiento de la línea de código, siempre irán después de la línea de código y dando un TAB antes de comenzar a escribir el comentario.

Correcto:

```
public int getArea(height,base){      //getting
      int area;
      res=base*height;
      return area;     //returning area
}
```

Incorrecto:

```
public int getArea(height,base){
//getting area
      int area;
      res=base*height;
      return area;
//returing area
```

c. Líneas en blanco

Se debe colocar un espacio en blanco entre una sección de código o de comentarios para mejorar la lectura.

Correcto:

```
if(condition1){
…
}     //next line is a white space

mainMenu(){
…
}
```

Incorrecto:

```
if(condition1){
…
}
mainMenu(){
…
}
```

d. Espacios en blanco

Se debe colocar un espacio en blanco entre las variables y operadores, con el fin de mejorar la lectura del código.

Correcto:

```
int x = 5 * 5 / 2;

String userData = getName() + nameData;
```

Incorrecto:

```
int x=5*5/2;

String userData=getName()+nameData;
```

e. Uso de paréntesis:

Usar paréntesis libremente para jerarquizar las expresiones

Correcto:

```
if((a == b) && (c == d));
```

Incorrecto:

```
if(a == b && c == d);
```

4. Formato de código

a. Bloques delimitados por llaves

La llave delimita las instrucciones del bloque debe ser colocada en líneas posterior a la que se declara esta.

Correcto:

```
If(x > 5)
{
     Int x = 5;
}
```

Incorrecto:

```
If(x > 5){
     Int x = 5;
}
```

b. Tamaño de línea

Evitar líneas de más de 80 caracteres

Correcto:

```
public int calculateArea(heigth, base){
      int result;
      result = base*heigth;
      return result;
```

Incorrecto:

```
public void calculateAreaFromAGemetricFigure(heigth,base,figuresName,figuresColor){
      int result;
      result = base*heigth;
      system.out.println("the figure"+figureName+"has the next área: "+result);
```

c. Identación:

Las identaciónes deben ser de 4 espacios, cada bloque de codigo interno tendrá un nivel extra de identación.

Correcto:

```
If(x < 5)
{
      Int x = 5;
}
```

Incorrecto:

```
If(x < 5)
{
                  Int x = 5;
}
```

d. Separación de líneas:

Si una expresión no entra en una línea, puede ser separada donde se localice una coma(“,”).

Correcto:

```
Public void function(int x, int y,
Int z);
```

Incorecto:

```
Public void function(int x, int
y,int z);
```

5. Declaraciones:

a. números por linea:

Una declaración por linea es recomendada dado que da oportunidad a que se coloque un cometario y es mas fácil de leer.

Correcto:

```
Int level; //identation level
Int size; //size of table
```

Incorrecto:

```
Int level, size;
```

b. Variables de retorno en funciones:

Las variables de retorno deben tener una variable calculada previamente.

Correcto:

```
return x;
```

Incorrecto:

```
return x + y / 2;
```

6. Convención de nombres

a. Variables:

Las variables deben contar con máximo 3 palabras usando **camelCasé.**

Correcto:

```
int i;
char cp;
float myWindth;
```

Incorrecto:

```
String nameFromTheMainUser;
Int numberOfItemsInInventory;
```

b. Constantes:

Las constantes debencontar con un máximo de 2 palabras, usando mayúsculas y separadas con un gion bajo (“_”);

Correcto:

```
int MIN_WIDTH = 4;
int MAX_WIDTH = 999;
```

Incorrecto:

```
int minWidth = 4;
int MAXWIDTH = 999;
```

c. Clases:

Las clases deben ser nombradas con pronombres y tener siempre su primera letra en mayúscula.

Correcto:

```
Public class Person;
```

Incorrecto:

```
Public class PERSON;
```

d. Métodos:

Los métodos deben ser verbos, usando Camel Casé.

Correcto:

```
public void run();
public void runFast();
public string getName();
```

Incorrectos:

```
public void Run();
public void RUNFAST();
public string Name();
```