Estándar de Codificación para Java
Departamento de Ciencias Computacionales
Escuela de Ingeniería y Tecnologías de Información
Campus Monterrey
Tecnológico de Monterrey

Antecedentes

Entre los principales propósitos de tener una convención de código están: entender el código más rápidamente; tener una visión consistente del código de manera que el lector pueda enfocarse en el contenido; facilitar el mantenimiento y aplicar las mejores prácticas del desarrollo.

Lineamientos

Los lineamientos de codificación son:

Tema	Lineamiento
Nombre del archivo que almacena la clase	El nombre del archivo que almacena un programa debe cumplir las siguientes reglas: • Debe ser un nombre descriptivo del propósito de la clase. • Debe iniciar con mayúscula y contener solamente letras y dígitos. No se debe emplear las letras "ñ" ó "Ñ" ni letras acentuadas. • Si requiere el uso de dos o más palabras, cada una debe iniciar con mayúscula. • Se debe utilizar la extensión **java**

	Ejemplos con estándar:
	`AppletJuego.java`
	`Raton.java`
	Ejemplos sin estándar:
	`aplicacion.java`
	`Applet_Área.java`
	`Aplicacion%Impuestos version 3.java`

Nombre del archivo que almacena datos de texto	El nombre del archivo que almacena datos debe cumplir con las siguientes reglas: • Debe ser un nombre descriptivo de los datos que contiene. • Debe iniciar con mayúscula y contener solamente letras y dígitos. No se debe emplear las letras "ñ" ó "Ñ" ni letras acentuadas. • Si requiere el uso de dos o más palabras cada una debe iniciar con mayúscula. • Se debe utilizar la extensión **txt**.

	Ejemplos con estándar:
	`Temperaturas.txt`
	`ListaAlturasMayores30.txt`
	Ejemplos sin estándar:
	`presiones.data`
	`datos.txt`
	`EnergíasCalorícas.dat`

Valores constantes	Una constante debe ser definida de acuerdo al valor que se desea utilizar, ya sea float, double, long, etc., de acuerdo a la tabla de constantes del anexo 1

Tema	Lineamiento

	Ejemplos con estándar:
	`3.0d`
	`7.5f`
	`-3`
	`20L`
	Ejemplos sin estándar:
	`12.5`
	`3.0`

Nombre de variables	El nombre de una variable debe cumplir con las siguientes reglas: • Debe estar compuesto por una letra minúscula que indique el tipo de valor que contendrá, seguida de una o más palabras que describan su contenido • La letra minúscula debe ser definida de acuerdo al tipo de dato, ver la tabla prefijos de variables del anexo 2. • Cada palabra de descripción debe iniciar con una letra mayúscula seguida por letras y dígitos. No se debe emplear la letra "ñ" ó "Ñ" ni letras acentuadas.

	Ejemplos con estándar:
	`dFuerza = 20.0d;`
	`dRadio = Double.parseDouble(sDatoLeido);`
	`iContador = 1L;`
	Ejemplos sin estándar:
	`ipeso = 25;`
	`edad = Integer.parseInt(sDatoLeido);`

Nombre de objetos	El nombre de un objeto debe seguir las siguientes reglas: • Debe iniciar con tres letras minúsculas que sirvan para identificar la clase a la que pertenece el objeto, seguidas de una o más palabras que describan su contenido • Cada palabra de descripción debe iniciar con una letra mayúscula seguida por letras y dígitos. No se debe emplear la letra "ñ" ó "Ñ" ni letras acentuadas.

	Ejemplos con estándar:
	`// se crea objeto de la clase Raton`
	`ratMickey = new Raton();`
	
	`// se mueve objeto de la clase Elefante`
	`eleDumbo.mueveDerecha();`
	Ejemplos sin estándar:
	`// se crea objeto de la clase Raton`
	`raton = new Raton();`
	
	`// se mueve objeto de la clase Elefante`
	`dumbo.mueveDerecha();`

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td rowspan="2">Nombre de arreglos de objetos</td><td>El nombre de un arreglo de objetos debe seguir las siguientes reglas:
Debe iniciar con seis letras minúsculas, las primeras tres que sirvan para identificar la clase a la que pertenece el objeto, las siguientes tres indican la estructura que contendrá, seguida de una o más palabras que describan su contenido
Si la estructura es vectorial, las tres letras deben ser ‘Arr’.
Si la estructura es matricial, las tres letras deben ser ‘Mat’.
Cada palabra de descripción debe iniciar con una letra mayúscula seguida por letras y dígitos. No se debe emplear la letra “ñ” ó “Ñ” ni letras acentuadas.</td></tr>
<tr><td>Ejemplos con estándar:<pre>// se crean los objetos del arreglo de la clase Raton
ratArrRatones = new Raton[3];

// se mueve cada objeto del arreglo de Elefantes
eleArrElefantes[iIndiceElefantes].mueveDerecha();</pre>Ejemplos sin estándar:<pre>// se crean los objetos del arreglo de la clase Raton
ratMatRatones = new Raton[3];

// se mueve cada objeto del arreglo de Elefantes
Elefantes[iIndiceElefantes].mueveDerecha();</pre></td></tr>
<tr><td rowspan="2">Nombre de variables vectoriales/ matriciales</td><td>El nombre de una variable que contiene una secuencia de valores organizados en vectores o matrices debe cumplir con las siguientes reglas:
Debe estar compuesto por cuatro letras minúsculas que indiquen el tipo de valores y la organización que contendrá seguida de una o más palabras que describan su contenido.
La letra minúscula debe ser definida de acuerdo a la tabla del anexo 2.
Si la estructura es vectorial, las siguientes tres letras deben ser ‘Arr’.
Si la estructura es matricial, las siguientes letras deben ser ‘Mat’.
Cada palabra de descripción debe iniciar con una letra mayúscula seguida por letras y dígitos. No se debe emplear la letra “ñ” ó “Ñ” ni letras acentuadas.
El conjunto de palabras debe ser descriptivo del dato que contiene y ser un sustantivo en plural.</td></tr>
<tr><td>Ejemplos con estándar:<pre>int[] iArrEdades = {22, 25, 53};
double[][] dMatTemperaturas = {{-2.0d, 25d}, {-53.5d, 0.0d}};</pre>Ejemplos sin estándar:<pre>int[] iArrE = {22, 25, 53};
double[][] dMTemperaturas = {{-2.0d, 25d}, {-53.5d, 0.0d}};</pre></td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td rowspan="2">Definición de constantes</td><td>Una constante debe ser definida de acuerdo a la siguiente regla:
Debe estar compuesto por una letra minúscula que indiquen el tipo de constante de acuerdo a la tabla de prefijos de variables y constantes del anexo 2, seguida de una o más palabras que describan su contenido.
Cada palabra de descripción debe iniciar con una letra mayúscula seguida por letras y dígitos. No se debe emplear la letra “ñ” ó “Ñ” ni letras acentuadas.
El conjunto de palabras debe ser descriptivo del dato que contiene
Una constante debe ser asignada a una variable del mismo tipo.
Una constante numérica solo especifica su signo cuando sea negativo.
Una constante con decimales, debe contener al menos un dígito antes del punto y al menos un decimal después del punto.</td></tr>
<tr><td>Ejemplos con estándar:<pre>final int iMAXIMA_TEMPERATURA = 100;
final double dMAXIMA_ALTURA = 250.0d;</pre>Ejemplos sin estándar:<pre>final int IMAXIMO = 4;
final double MAYOR = 50;</pre></td></tr>
<tr><td>Orden de los elementos de una clase</td><td>Todas las clases deben contener la siguiente estructura:
imports necesarios
Comentarios de inicio de la clase
Encabezado de la clase
Constructores
Métodos de acceso y métodos de modificación
Otros Métodos restantes con su documentación previa incluida
Cierre de la clase</td></tr>
<tr><td rowspan="2">Comentarios de inicio de la clase</td><td>Los comentarios de inicio deben estar dentro de un bloque formado por los caracteres ‘/*’ al inicio y ‘*/’ al final y contener la siguiente información: nombre de la clase, descripción, autor/autores, fecha y versión, cada línea debe empezar con un carácter ‘*’ alineado al primero, así como el último que cierra el bloque de comentarios.
Para describir el autor, la fecha y la versión, utilizar los tags @author, @date y @version.</td></tr>
<tr><td>Ejemplo con estándar:<pre>/*
 * AppletAnimacion
 *
 * Applet que permite al usuario usar el teclado para
 * animar el movimiento de un objeto
 *
 * @author Juan Pérez   A00999999
 * @date 2/2/2016
 * @versión 1.0
 /</pre>Ejemplo sin estándar:<pre>/ Tarea
 * Juan Pérez
 */</pre></td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td rowspan="2">Encabezado de la clase</td><td>Declaración inicial de la clase, con su respectiva herencia e implementaciones si es requerido, seguido de un espacio y la llave que abre la clase.</td></tr>
<tr><td>Ejemplo con estándar:
<pre>public class Juego extends Applet implements Runnable {</pre>Ejemplo sin estándar:
<pre>public class Juego extends Applet implements Runnable
{</pre></td></tr>
<tr><td rowspan="2">Comentarios de constructores y métodos</td><td>Los comentarios de los constructores y métodos deben estar dentro de un bloque formado por los caracteres ‘/*’ al inicio y ‘*/’ al final y contener la siguiente información: nombre del constructor o método, descripción, parámetros, valor de retorno y excepciones, en caso de que maneje alguno de los tres anteriores, cada línea debe empezar con un carácter ‘*’ alineado al primero, así como el último que cierra el bloque de comentarios.
Para describir cada parámetro, valor de retorno y excepción, utilizar los tags @param, @return y @exception.</td></tr>
<tr><td>Ejemplo con estándar:
<pre>/*
 * calculaArea
 *
 * obtener el área de una circunferencia dado su radio
 *
 * @param dRadio es el valor <code>double</code> del radio
 * @return un valor <code>double</code> que representa el área
 */</pre>Ejemplo sin estándar:
<pre>/*
 * calculaArea
 *
 * obtener el área de una circunferencia dado su radio
 *
 */</pre></td></tr>
<tr><td>Constructor</td><td>Un constructor debe ser definido igual que el nombre de la clase y puede o no contener parámetros, de acuerdo a la siguientes reglas:
• Debe iniciar con la palabra reservada public
• A continuación se escribe un espacio y el nombre de la clase
• Si hay parámetros estos se definen entre paréntesis, dejando un espacio después del carácter ‘,’ que separa cada parámetro. Si no hay parámetros solo se escriben los paréntesis sin espacio entre ellos.
• A continuación se debe dejar un espacio y escribir el carácter ‘{’
Posteriormente se encuentran las instrucciones del constructor y al terminar se usa el carácter ‘}’ en una sola línea que esté alineado al lado izquierdo del encabezado del método.</td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td rowspan="2">Método</td><td>El encabezado de un método debe cumplir las siguientes reglas:
Debe iniciar con la palabra reservada public o private o protected.
A continuación se define el tipo de dato que se obtendrá o en su defecto la palabra reservada void.
Después se escribe un espacio y el nombre del método que debe ser una o más palabras que describan su contenido, donde la primer palabra debe estar en minúsculas y si se requiere dos palabras o más éstas deben iniciar con mayúscula.
A continuación se escriben paréntesis que contengan los parámetros si es que se requieren.
Después se debe escribir un espacio y la palabra reservada throws y el nombre de la excepción, si el método es capaz de arrojar un error de ejecución
Se deberá dejar un espacio y escribir el carácter ‘{‘
A continuación se encuentran las instrucciones del método y al terminar se usa el carácter ‘}’ en una sola línea que esté alineado con el lado izquierdo del encabezado del método.</td></tr>
<tr><td>Ejemplo con estándar:<pre>
public int calculaArea(int iRadio) {
    return Math.PI * Math.pow(iRadio, 2.0d);
}
</pre>Ejemplo sin estándar:<pre>
public int calculaArea(int iRadio)
{
    return Math.PI * Math.pow(iRadio, 2.0d);
}
</pre></td></tr>
<tr><td>Comentarios internos</td><td>Se recomienda agregar comentarios internos a las instrucciones o bloques de instrucciones para facilitar su comprensión.
Los comentarios deben cumplir las siguientes reglas:
Si es una línea deben iniciar con // , o entre /* ….. */
Si es mas de una línea de comentarios deben estar entre /* ….. */, como se maneja el comentario de un constructor o método</td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td></td><td>Ejemplos con estándar:

```
// el objeyo camina hacia la derecha y arriba 2 pixeles
setX(getX() + 2);
setY(getY() + 2);

/*  Se revisa si el dibujo esta saliendo debajo el fondo del
applet
     para posicionarlo en el fondo  */
if (ratMickey.getY() + ratMickey.getHeight() > getHeight()) {
     ratMickey.setY(getHeight());
}
```

Ejemplos sin estándar:

```
// se mueve
setX(getX() + 2);
setY(getY() + 2);

/*  Se checa colision
      y se reposiciona */
if (ratMickey.getY() + ratMickey.getHeight() > getHeight()) {
     ratMickey.setY(getHeight());
}
```
</td></tr>
<tr><td rowspan="2">Espacios en Blanco</td><td>Las piezas del programa se deben escribir con suficiente espacio para que no aparezcan amontonadas. Por lo tanto se deben cumplir los siguientes lineamientos:
• Dejar un espacio en blanco entre cada operador.
• Dejar una línea en blanco antes de cada bloque de instrucciones con un propósito.</td></tr>
<tr><td>Ejemplo con estándar:

```
// obtengo el valor real de cada dato pedido de los coeficientes
dA = Double.parseDouble(sDatoA);
dB = Double.parseDouble(sDatoB);
dC = Double.parseDouble(sDatoC);

// calculo el discriminante
dDiscriminante = Math.pow(dB, 2.0d) - 4.0d * dA * dC;
dRaiz1 = (-dB + Math.sqrt(dDiscriminante)) / (2.0 * dA);
dRaiz2 = (-dB - Math.sqrt(dDiscriminante)) / (2.0 * dA);
```

Ejemplo sin estándar:

```
// obtengo el valor real de cada dato pedido de los coeficientes
dA=Double.parseDouble(sDatoA);
dB=Double.parseDouble(sDatoB);
dC=Double.parseDouble(sDatoC);
// calculo el discriminante
dDiscriminante = Math.pow(dB,2.0d)-4.0d*dA *dC;
dRaiz1 = (-dB + Math.sqrt(dDiscriminante)) /(2.0*dA);
dRaiz2 = (-dB - Math.sqrt(dDiscriminante)) /(2.0*dA);
```
</td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td rowspan="2">Sangría</td><td>Las instrucciones deben estar alineadas con una sangría creada con la tecla Tab para mostrar la jerarquía de las instrucciones por bloque, todas las instrucciones de control deben tener llaves que delimiten sus instrucciones, la llave que abre “{“ debe ponerse con un espacio despues de la instrucción de control, y cuando se termine de escribir con sangría las instrucciones que contiene esta instrucción de control, deberá escribirse la llave que cierra “}”, alineada al lado izquierdo de la instrucción de control que está delimitando.</td></tr>
<tr><td>Ejemplo con estándar:
<pre>
while (iContador < 10) {
   if (iContador % 2 == 0) {
       lTotal = lTotal + iContador;
   }
   iContador = iContador + 2;
}
</pre>Ejemplo sin estándar:
<pre>
while (iContador < 10) {
 if (iContador % 2 == 0)
    lTotal = lTotal + iContador;
 iContador = iContador + 2;
}
</pre></td></tr>
<tr><td rowspan="2">Número de instrucciones por línea</td><td>Sólo debe haber una instrucción por renglón.</td></tr>
<tr><td>Ejemplo con estándar:
<pre>
if (dAngulo == 45.0d) {
      sTitulo= “Dentro”;
}
else {
      sTitulo= “Fuera”;
}
</pre>Ejemplo sin estándar:
<pre>
if (dAngulo == 45.0d) sTitulo = “Dentro”;
else sTitulo= “Fuera”;
</pre></td></tr>
<tr><td>Comparación con boleanos</td><td>Al utilizar en una condición una variable boleana, o una expresión boleana, no deberá de compararse con las constantes boleanas true o false.</td></tr>
</table>

<table>
<tr><th>Tema</th><th>Lineamiento</th></tr>
<tr><td></td><td>Ejemplo con estándar:

```
if (bColisiono) {
      iNivel --;
      dVelocidad = 0;
}
else {
      dVelocidad += 50;
}
```

Ejemplo sin estándar:

```
if (bColisiono == true) {
      iNivel --;
      dVelocidad = 0;
}
else {
      dVelocidad += 50;
}
```

</td></tr>
<tr><td rowspan="2">Comparación con boleanos para retornar resultado de un método</td><td>Cuando tengamos un método que debe retornar un valor boleano dependiendo de una condición, no utilizar el estatuto if, retornar directamente la condición boleana.</td></tr>
<tr><td>Ejemplo con estándar:

```
  public boolean estaChocado() {
     return iPosicion > iMAXIMA_POSICION;
 }
```

Ejemplo sin estándar:

```
public boolean cruzaFrontera() {
     if (iPosicion > iMAXIMA_POSICION) {
           return true;
     }
     else {
           return false;
     }
 }
```

</td></tr>
<tr><td>Mejores prácticas</td><td>El código deberá cumplir con lo siguiente:
• No dejar variables en el código sin utilizar.
• Cada variable utilizadas en el código debe tener un valor inicial apropiado.
• No utilizar paréntesis de más en una expresión.
• Cada método debe tener un propósito específico para hacer el código modular (pedir un valor, calcular un resultado, desplegar información).
• No usar break como instrucción en el bloque de instrucciones de un ciclo.</td></tr>
</table>

Referencias bibliográficas

PSP(sm): A Self-Improvement Process for Software Engineers Hardcover –Watts S. Humphrey

Java Code Convention
http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Documento elaborado por:
Comité del Área de Programación, Ciencias Computacionales, Campus Monterrey.
Profesor responsable: Ing. Antonio Mejorado.

Anexo 1

Tabla de valores constantes

Tipo de dato	Valores default
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Anexo 2

Tabla de prefijos de variables y constantes

Tipo de dato	Prefijo
int	i
long	l
float	f
double	d
char	c
String	s
boolean	b