UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Proyecto "*Sistema de Gestión de Reservas de Aulas y Laboratorios Universitarios*"

Curso: *SI-982 Programación Web II*

Docente: *Mag. Patrick Cuadros Quiroga*

Integrantes:

Valverde Zamora, Jean Pier Elias ***2020066920***

Lizárraga Pomareda, Sergio ***2020066921***

Tacna – Perú
2025

Sistema de Reserva de Aula y Laboratorios
Documento de Estándares de Programación
Versión 1.0

Historia de Revisión

Historial de revisiones				
Ítem	**Fecha**	**Versión**	**Descripción**	**Equipo**
1		1.0	Versión Final.	

Tabla de Contenidos

Estándares de Programación

1. OBJETIVO

Reglamentar la forma en que se implementará el código fuente del proyecto, pasando, por las variables, controles, clases, métodos, ficheros, archivos y todo aquello que esté implicado en el código,

Mejorar y uniformizar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

- Los nombres de variables serán mnemotécnicos con lo que se podrá saber el tipo de dato de cada variable con sólo ver el nombre de la variable.
- Los nombres de variables serán sugestivos, de tal forma que se podrá saber el uso y finalidad de dicha variable o función fácilmente con solo ver el nombre de la variable.
- La decisión de poner un nombre a una variable o función será mecánica y automática, puesto que seguirá las reglas definidas por nuestro estándar.
- Permite el uso de herramientas automáticas de verificación de nomenclaturas.

Por tanto, se seguirán dichos patrones para un entendimiento legible del código y para facilitar el mantenimiento del mismo.

2. DECLARACION DE VARIABLES

Se propone que la declaración de las variables, se ajusten al motivo para la que se requieran. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente:

- La longitud debe ser lo más recomendable posible. No debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 16 caracteres.
- Alcance de la variable

  A medida que aumenta el tamaño del proyecto, también aumenta la utilidad de reconocer rápidamente el alcance de las variables. Esto se consigue al escribir un prefijo de alcance de una letra delante del tipo de prefijo propio, sin aumentar demasiado la longitud del nombre de las variables.

Alcance	Prefijo	Ejemplo
Global	G	GPuntaje
Público	P	Ptiempo_restante
Privado	Pr	prCantidadVenta

- El tipo de dato al que pertenece la variable.

  Por lo tanto la estructura de la variable es como sigue:

Estructura	Descripción de la Variable
LONGITUD. MAX.	□ 1 □□ 16 □
FORMATO	*Minúscula la primera parte y luego la segunda con Mayúsculas*
EJEMPLO	numCuenta

Siendo el nombre que identifica a la variable: **numCuenta**

2.1 Descripción de la Variable.

Nombre que se le asignará a la variable para que se le identifique y deberá de estar asociada al motivo para la cual se le declara.

numPuntaje: puntuación del usuario?

numtiempo_restante: temporizador del juego ejecutado?

Colores: Pieza fundamental del minijuego

Ejemplos: idCuenta, tipoEstado, instalacion

2.2 Variables de Tipo Arreglo

En el caso de las definiciones de arreglos de elementos se declarará la variable con el prefijo de “lista”, el cual nos dará entender que se trata de una variable del tipo arreglo la cual contendrá de cero a mas datos, según el tamaño declarado.

Ejemplos: listaTiposDoc
listcolores tamaño max: 1 a 15

'Red','Blue','Green','Pink','Black','Yellow','Orange','White','Purple','Brown'

3. Definición de Controles

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo al cual pertenece y la función que cumple dentro de la aplicación.

3.1 Tipo de datos

Tipo de variable	Mnemónico	Descripción
Integer	in	Entero de 32 bits con signo.
String	st	Cadena de caracteres

3.2 Prefijo para el Control

- Label, estará asociado al prefijo lbl
- Entry, estará asociado al prefijo txt
- Button, estará asociado al prefijo btn

3.3 Nombre descriptivo del Control

Formado por la descripción de la función que lleva a cabo el control, esta debe ser descrita en forma específica y clara.

Tipo de control	Prefijo	Ejemplo
Label	lbl	lblInstrucciones
TextBox	txt	txtRespuesta
Button	btn	btnLogin

3.4 Declaración de variables, atributos y objetos

1. Se debe declarar una variable por línea.

Título	Descripción
Sintaxis	[int] [puntaje]
Descripción	
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public int puntaje Indica una variable o atributo que guardará un nombre.

Título	Descripción
Sintaxis	[Int] [tiempo_restante]
Descripción	Permite implementar una duración para el juego de 30 segundos Inicia en el código con un valor de 0 ”tiempo_restante=0”
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public int tiempo_restante Indica una variable o atributo que guardará un nombre.

Título	Descripción
Sintaxis	[String] [respuesta]
Descripción	Permite escribir el nombre del color que aparece,
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public int tiempo_restante Indica una variable o atributo que guardará un nombre.

Título	Descripción
Sintaxis	[TipoVariable] [Nombre de la Variable]
Descripción	Todas las variables o atributo tendrán una longitud máxima de 30 caracteres. El nombre de la variable puede incluir más de un sustantivo los cuales se escribirán juntos. Si se tuvieran variables que puedan tomar nombres iguales, se le agregará un número asociado (si está dentro de un mismo método será correlativo).
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public int puntaje Indica una variable o atributo que guardará un nombre.

3.5 Declaración de clases

Título	Descripción
Sintaxis	[Tipo] Class [Nombre de Clase]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. Tipo se refiere a si la clase será: Private, Public o Protected.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Private Class Empleado Indica una clase Empleado

3.6 Declaración de métodos

Título	Descripción

Sintaxis	nombreProcedim[(ListaParámetros)]
Descripción	El nombre del método constará hasta de 25 caracteres. La primera letra de la primera palabra del nombre será escrita en minúscula y las siguientes palabras empezarán con letra mayúscula.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Protected calcularSueldo(String empleado) Indica un método calcularSueldo que recibe una variable por valor de tipo string al ámbito de la clase

3.7 Declaración de funciones

Título	Descripción
Sintaxis	[string] inicio[(iniciar)]
Descripción	si tiempo_restante es igual a 30 dara inicio a la funcion timer() iniciar la funcion siguiente_color
Observaciones	En la declaración de objetos no se deberá utilizar caracteres como: ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _.
Ejemplo	Public string (iniciar) Indica una función dará inicio al programa

Título	Descripción
Sintaxis	[int] timer[()]
Descripción	mientras que tiempo_restante es mayor a 0 se restara -1
Observaciones	En la declaración de objetos no se deberá utilizar caracteres como: ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _.
Ejemplo	Global int (timer) Indica una función de una cuenta hacia atrás

Título	Descripción
Sintaxis	[string] siguiente_color[()]
Descripción	si tiempo_restante es mayor a 0 limpie el cuadro respuesta si respuesta es igual a colores se suma +1 a puntaje se borra la respuesta y randomiza colores para empezar otra vez hasta que tiempo_restante sea igual a 0
Observaciones	En la declaración de objetos no se deberá utilizar caracteres como: ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _.
Ejemplo	Public string (siguiente_color) Indica una funcion que es el juego en si

3.8 Control de versiones de código fuente

Cada modificación realizada será guardada de la forma:

Título	Descripción
Formato	[NOMBRE DOCUMENTO][_][FECHA][_][HORA] donde y la fecha estará en formato yyyymmdd y la hora en formato HHMM.
Descripción	Se generarán archivos con las siguientes extensiones:.zip o .rar. Por ejemplo: WSTENNIS_20070421_2056.zip

4. Métodos, Procedimientos y Funciones definidos por el Usuario.

def inicio(iniciar):

iniciarJuego

Verbo: iniciar
Sustantivo: Juego

def siguiente_color():

mostrarColor

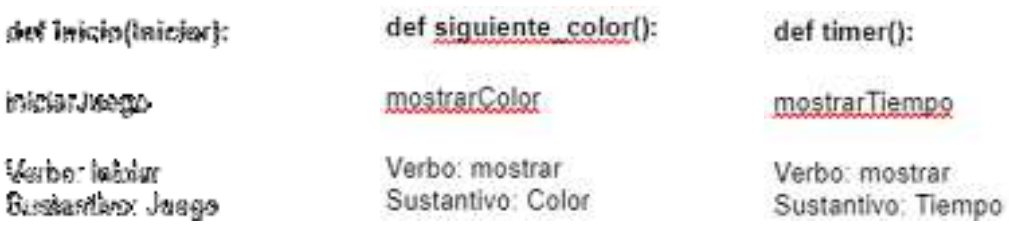
Verbo: mostrar
Sustantivo: Color

def timer():

mostrarTiempo

Verbo: mostrar
Sustantivo: Tiempo

Métodos, Procedimientos y Funciones definidos por el Usuario.



5. Beneficios

- La documentación hace más legible un programa.
- Al documentar bien un programa desde un principio se evita que para cada modificación deba estudiarse profundamente el funcionamiento del programa, redescubriendo todo lo

no documentado, con la ventaja adicional de que generalmente quién modifica el programa no es siempre quién lo escribió.
- Facilita la reutilización de módulos y rutinas desde cualquier otro programa o el mismo.
- Ayuda a determinar cuándo debe ser reescrito un código. Si existen problemas para explicar el código con un comentario, probablemente el código esté mal escrito.

6. Conclusiones

Después de realizar el presente trabajo de investigación, se presentan las siguientes conclusiones:

- Los diccionarios son un buen elemento de manipulación cuando se implica un problema de traducción entre idiomas.
- Para poder ordenar de manera aleatoria un diccionario, se debe transformar a una tupla y de esta manera recién poder usar librerías.