

PROGRAMACIÓN II

Angel Chuncho / angel.chuncho@epn.edu.ec


Taller #01 y #02

Taller #01

En la clase se desarrolló el taller #01 denominado "Números al Infinito" que consistía en formar grupos de 10 integrantes y cada integrante se debía colocar un número del 0 al 9 con la finalidad de armar los números que dictaba el profesor.

Taller #02

Para la ejecución del taller #02 se necesitó una funda de malvaviscos y fideos tallarín, el objetivo era realizar la torre más alta pero empleando la menor cantidad de recursos posibles.

 Figura 1. Desarrollo de los talleres

Entorno de Desarrollo Integrado (IDE) / Visual Studio Code

Existen IDE's locales como IDE's en la nube, entre los ejemplos de IDE's se tiene: Visual Studio Code, NetBeans, IntelliJ, etc. Entre las características de los IDE's se tiene:

- Entorno de desarrollo estandarizado
- Independencia de plataformas / SO
- Mejor rendimiento
- Personalización

El IDE que se utilizara en este semestre es Visual Studio Code, además se debe descargar e instalar lo siguiente:

- JDK v17
- Extension Pack for Java
- Project Manager for Java
- Makefile Tools
- Markdown PDF
- markdownlint
- Draw.io Integration
- Draw.io Preview

 Figura 2. Herramientas a utilizar

Atajos de Teclado en Visual Studio Code

1. Command Palette → CTRL+SHIFT+P
2. Quick Open → CTRL+P
3. Toggle Sidebar → CTRL+B
4. Multi-Selector Cursor → CTRL+D
5. Copy Line → CTRL+ALT+UP or SHIFT+ALT+DOWN
6. Comment Code Block → SHIFT+ALT+A(Multi-line comment) or CTRL+K+C(Single-line comment)
7. Go back/ move forward
→ ALT+→
8. Show All Symbols → Ctrl+T
9. Trigger suggestion and Trigger parameter hints → Ctrl+SPACE, Ctrl+Shift+Space

Markdown

Es un lenguaje de marcado con el cual se agrega formato a documentos de texto plano. Los archivos utilizan la extensión .md, .markdown o .mdx; algunos comandos útiles son:

titulo# ## ### #### #####

palabras en negrita

palabras en cursiva

palabras en negrita y cursiva ==texto resaltado==

~~texto tachado~~

 Figura 3. Comandos markdown

GitHub

Creado en 2005 por el mismo Linus Torvalds (creador de Linux) como herramienta para facilitar el desarrollo colaborativo de software. Algunos comandos utilizados son:

Para version de git

\$git --version

Para identidad

\$git config --global user.name "inserte su nombre de usuario de git"

\$git config --global user.email "inserte su correo de git"

Para verificar

\$git config user.name

\$git config user.email

Control de versiones

\$git init

\$git status

\$git add

\$git add NombreCarpeta/NombreArchivo \$git commit -m 'mensaje'

Conceptos Basicos

El nombre del archivo debe ser el mismo que el nombre de la clase y en mayusculas

Código: Escrito por los programadores

Compilación: Esta compilación devuelve un Bytecode

Bytecode: Son instrucciones para la java Virtual Machine

JVM: La java Virtual Machine interpreta el bytecode

Multiplataforma: Se ejecuta en diferentes sistemas operativos

Algoritmia

Pseudocódigo

Diagramas de flujo Código (debugging)

Detención de errores

Estructuras de Control

For

While

Do while

Tipos de Datos

byte: numérico entero con signo

short: numérico entero con signo

int: numérico entero con signo

float: numérico en coma flotante double: numérico en coma flotante

char: carácter unicode boolean: dato lógico

void: vacío

Documentación POO


Es útil para mantener la claridad de nuestro código y que pueda ser interpretado por varias personas.

```
/** Clase con ejemplos de testing de java*
 * @author: pat_mic
 * @version: 1.0
 * @see<a href= "http://www.abc.com" /> pat_mic_github</a>*/
public class Testing{    /** valida la compilación del java*/
    public static void TestJava(){
        System.out.print("java are ready..!");
    }
    /** valida la compilación del java con saludo
    * @param nombre: escribe tu nombre
    * @param fecha: es la fecha actual
    * @return un saludo*/
    public static String TestJavaSaluda(String nombre, String fecha){
        return "java te saluda "+nombre+" "+fecha;
    }
}
```

```
}
}
```

Definiciones POO

- **Objeto:** un objeto es ya una entidad concreta que se crea a partir de la plantilla que es la clase.
- **Clase:** una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de determinado tipo.
- **Atributo:** características que aplican al objeto solo en el caso en que el sea visible en pantalla por el usuario
- **Métodos:** funciones que permite efectuar el objeto y que nos rinden algún tipo de servicio durante el transcurso del programa.

Nota: En clase se realizó un pequeño taller que consistía en abstraer los atributos y métodos que podían tener una iguana de juguete y una araña de juguete.  Figura 4. Taller en clase

Herencia

Es un mecanismo por el que se pueden crear nuevas clases a partir de otras existentes, heredando y posiblemente modificando y/o añadiendo operaciones y posiblemente añadiendo atributos. También se denomina extensión o generalización. Al extender una clase, se heredan todas las operaciones del padre y se pueden añadir nuevas operaciones.

```
public class Coche extends Vehiculo{
    //cilindrada del coche
    private int cilindrada
    //retorna la cilindrada del coche
    public int cilindrada(){
        return cilindrada;
    }
    //cambia la cilindrada del coche
    public void cambiaCilindrada(int c){
        this.cilindrada = c;
    }
}
```

Una subclase puede redefinir ("override") una operación en lugar de heredarla directamente, se indica mediante @Override

```
public class Coche extends Vehiculo{
    //cilindrada del coche
    private int cilindrada
    /*
     * Construye un coche
     * @param color color del coche
     */
}
```

```
        * @param numSerie número de serie del coche
        * @param cilindrada del coche
    */
    public Coche(Color color, int numSerie, int cilindrada){
        super(color, numSerie);
        this.cilindrada = cilindrada;
    }
    //Obtiene la cilindrada del coche
    public int cilindrada(){
        return cilindrada;
    }
    //cambia la cilindrada del coche
    public void cambiaCilindrada(int c){
        this.cilindrada = c;
    }
    @Override
    public String toString(){
        return super.toString() + ", cilindrada = " + cilindrada;
    }
}
```

Modificadores de Acceso para Miembros de Clases


- `:` accesible desde el paquete
- `public`: accesible desde todo el programa
- `private`: accesible sólo desde esa clase
- `protected`: accesible desde sus subclases y, en Java, desde cualquier clase en el mismo paquete.


Diagramas de Caso de Uso

Diagrama de comportamiento en lenguaje UML, con la que se representan procesos empresariales, así como sistemas y procesos de programación orientada a objetos.

Se compone de diagramas, actores, funcionalidades, relaciones, especificación.

Relaciones

- **include**: relación de inclusión indica que un caso base incorpora explícitamente el comportamiento de otro caso de uso.
- **extend**: relación de extensión indica que un caso de uso base incorpora implícitamente el comportamiento de otro caso de uso. Figura 4. Taller en clase

UML La palabra UML significa Lenguaje Unificado de Modelado (UML), es útil para representar el programa de código a una manera más visual utilizando diagramas y figuras tanto en estructura como en comportamiento. Figura 5. Taller en clase

Interfaces

POO e Interfaz de Usuario

Interfaz o fachada

Es un elemento en la programación que permite especificar un conjunto de operaciones en clase que pueden ser utilizadas por otras clases, en resumen permite definir operaciones o actividades a una clase.

Es importante recalcar que una interfaz no es una clase puesto que no tiene atributos y solo define un conjunto de operaciones, más bien se acopla (implementa) a una clase.

Características:

- No puede crear instancias de variables y crear un objeto.
- No puede contener métodos concretos.
- Solo utiliza el especificador público (public).

User Interface (UI): espacio donde interactúan los seres humanos y máquinas. Permite el funcionamiento y control de la máquina desde el humano.

Tipos de interfaces de usuario (UI)

- **Interfaz de línea de comandos (CLI):** interfaz alfanumérica que solo presenta texto.
- **Interfaz gráfica de usuario (GUI):** representa gráficamente los elementos de control y medida.
- **Interfaz natural de usuario (NUI):** pueden ser táctiles, trabajar mediante reconocimiento del habla o movimientos corporales.
- **Interfaz natural de usuario (NUI):** pueden ser táctiles, trabajar mediante reconocimiento del habla o movimientos corporales.
- **Interfaz física o hardware:** teclados, ratones, pantallas, etc.
- **Interfaz lógica o software:** programas, vistas, menús, formularios, etc.

User Experience (UX): se refiere al aspecto emocional del usuario, es decir, como se siente el usuario al experimentar con la interfaz.

vs

Usabilidad: se refiere a la sencillez que presenta una página para su uso, facilitando la navegación para el usuario.

Código de ejemplo:

A continuación, se presenta un ejemplo de interfaz gráfica utilizando el proyecto Tinder Pet Lover

```
import java.awt.Color;
import javax.swing.JFrame;
public class NewApp{
    public static void main(String[]args) throws Exception{
        JFrame frm = new JFrame(); //Crea el
frame
        frm.setTitle("Principal"); //Coloca el
título al frame
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Salir de la
aplicación
        frm.setResizable(false); //Evita que
el frame sea cambiado de tamaño
        frm.setSize(420,420); //Coloca el
tamaño del frame
```

```
        frm.setVisible(true); //Hace
visible al formulario
        // frm.lconImage = new ImageIcon("logo.png"); //Crea un
ImageIcon
        // frm.setlconImage(image.getImage()); //Cambia el
icono del frame
        // frm.getContentPane().setBackground(new Color(12,45,54)); //Cambia el
color de fondo
    }
}
```

Se visualiza:



Figura 6. Interfaz inicial de Tinder Pet Lover