

## **Taller Training de Calidad**

**Realizado por:** Alejandro Mesa Ramírez

**Correo:** [amesaramirez@outlook.es](mailto:amesaramirez@outlook.es)

### **1. Defina en sus palabras que es la calidad de software**

Es la parte de las ingenierías relacionadas con “TICs” encargada de proporcionar confianza en el que los requisitos o solicitudes que proponen los clientes serán cumplidos en su totalidad. En palabras más simples es el proceso que hace parte en cada ciclo de desarrollo de software desde que ingresan las historias de usuario hasta que llega al proceso de aceptación por parte del cliente. Se encarga de verificar y establecer parámetros o indicadores que permitan medir el nivel de calidad en aspectos como funcionalidad, seguridad, compatibilidad, entre otros y cuyo objetivo es detectar defectos y aportar información en la toma de decisiones del producto para poder ofrecer un resultado confiable y a gusto a la solicitud del cliente.

### **2. Explique la diferencia entre un sistema de control de versiones centralizado y un sistema de control de versiones distribuido**

El sistema de control de versiones centralizado permite la comunicación con desarrolladores en otros sistemas teniendo un único servidor donde descargan la última versión del software cuando lo necesitan mientras que en un sistema distribuido a pesar de que cumple la misma función de descargar la copia actualizada del software, este, replica todo el repositorio lo que genera un consumo de disco local importante a considerar.

### **3. ¿Cuál es el comando utilizado en Git para clonar un proyecto?**

git clone [enlacerepositorio](#)

### **4. ¿Qué realizan los siguientes comandos de git?**

- git add -miClase: No hace nada.
- git add -A: Guarda los cambios no solo en la raíz donde coincide con algún <path>, sino también donde los índices ya tienen una entrada. En palabras simples agrega todos los cambios que hicimos a los archivos y carpetas de nuestro repositorio al área de preparación de Git.
- git status: Verificar el estado actual del repositorio, en qué rama está y si hay algo pendiente para hacer “commit”.
- git pull: Actualiza el repositorio local con lo que haya en el repositorio remoto.
- git push: Subir cambios que se le hayan hecho “commit” del repositorio local al repositorio remoto.

- `git commit -m "Hola"`: Para tomar todos los cambios que hayan sido agregados recientemente y el `-m` es necesario para poner un mensaje e identificarlo en el repositorio, este mensaje siempre va entre comillas.
- `git log`: Muestra los logs de los "commit"

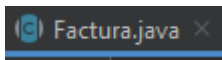
### 5. Describa en desarrollo de software que es la herencia y que es el polimorfismo

- La herencia en el desarrollo de software es cuando un objeto "hereda" las propiedades y métodos del objeto padre, permitiendo agregar, modificar o quitar elementos de utilidad en el objeto hijo según lo vea conveniente, además, permite organizar los objetos como una jerarquía empezando desde lo mas general hacia lo mas específico. Lo que ofrece una buena opción para ahorrar código.
- El polimorfismo es la habilidad de un objeto de tomar muchas formas y permite ignorar los tipos de datos y ejecutar las operaciones

#### Ejemplo:

**Nota:** Este código no es de mi autoría, es un ejemplo que me enseñaron en la universidad y me ayudo a entender los conceptos.

Clase padre:



```
//Implementamos la clase padre
public abstract class Factura {
    //Declaramos variables
    private String id;
    private double importe;

    //Declaramos métodos accesorios
    public void setId(String id) {
        this.id = id;
    }

    public String getId() { return this.id; }

    public void setImporte(double importe){
        this.importe=importe;
    }

    public double getImporte() { return this.importe; }

    //Declaramos un método abstracto
    public abstract double calcularTotal();
}
```

Clase hijo 1:

```
Factura.java
//Declaramos clase hijo y heredamos

public class FacturaConIva extends Factura {

    //Implementamos metodos abstractos
    @Override
    public double calcularTotal() {
        //Modificamos el método a gustos para añadir importe
        return this.getImporte()*1.21;
    }
}
```

Clase hijo2:

```
FacturaSinIva.java
//Declaramos clase hijo2 y heredamos

public class FacturaSinIva extends Factura {
    //Implementamos metodos abstractos
    @Override
    public double calcularTotal() {
        return this.getImporte();
    }
}
```

Clase Main:

```
principal.java
//Declaramos la clase principal
public class principal {
    //Declaramos el método main
    public static void main(String [] args){
        //Creamos objetos y establecemos valores
        FacturaConIva f1=new FacturaConIva();
        f1.setId("1");
        f1.setImporte(100);
        FacturaSinIva f2=new FacturaSinIva();
        f2.setId("2");
        f2.setImporte(100);

        //Imprimimos salida
        System.out.println("Importe con iva: "+ f1.calcularTotal());
        System.out.println("Importe sin iva: "+ f2.calcularTotal());
    }
}
```

Podemos apreciar que creamos objetos a partir de los hijos, pero estos acceden a la variables y métodos que fueron declarados en el padre.

Resultado:

```
Run: principal x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Importe con iva: 121.0
Importe sin iva: 100.0
Process finished with exit code 0
```

Ahora para apreciar el polimorfismo creamos un método estático:

```
public static void imprimirImporte(Factura f) { System.out.println("Importe:" + f.calcularTotal()); }
```

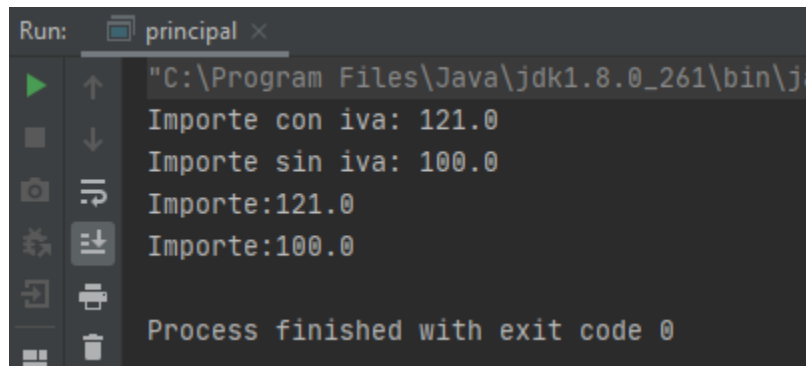
Y agregamos al main:

```
public class principal {
    //Declaramos el método main
    public static void main(String [] args){
        //Creamos objetos y establecemos valores
        FacturaConIva f1=new FacturaConIva();
        f1.setId("1");
        f1.setImporte(100);
        FacturaSinIva f2=new FacturaSinIva();
        f2.setId("2");
        f2.setImporte(100);

        //Imprimimos salida
        System.out.println("Importe con iva: " + f1.calcularTotal());
        System.out.println("Importe sin iva: " + f2.calcularTotal());

        imprimirImporte(f1);
        imprimirImporte(f2);
    }
}
```

Resultado:

A screenshot of a Java IDE's Run console. The window title is "Run: principal x". The console shows the execution path: "C:\Program Files\Java\jdk1.8.0\_261\bin\java.exe". The output consists of four lines: "Importe con iva: 121.0", "Importe sin iva: 100.0", "Importe:121.0", and "Importe:100.0". At the bottom, it states "Process finished with exit code 0". On the left side of the console, there is a vertical toolbar with icons for running, stepping through, and other debugging actions.

```
Run: principal x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe"
Importe con iva: 121.0
Importe sin iva: 100.0
Importe:121.0
Importe:100.0
Process finished with exit code 0
```

Podemos apreciar que sin importar el tipo de datos este ejecuta la operación lo que es un principio en el polimorfismo y puede ayudar a reducir código y simplificar procesos para que no sean dependientes de algo en específico.