

# Pontificia Universidad Católica del Ecuador – PUCE TEC

**Carrera:** Tecnología en Desarrollo de Software

**Materia:** Programación 2 (NRC: 4369)

**Docente:** Danni Brito

**Fecha:** 28 Junio de 2025

**Nombre del estudiante:** Cristian Pareja

## INSTRUCCIONES GENERALES

- Responde todas las preguntas de forma clara y concisa
- Utiliza ejemplos cuando sea necesario para respaldar tus respuestas
- Escribe con letra legible
- Tiempo estimado: 60 minutos
- Se permite el uso de apuntes personales, pero **no está permitido el uso de IA generativa** (como ChatGPT).
- Calificación total: 100 puntos

### ♥ Reflexión breve:

Esta evaluación me ayuda a saber si estoy enseñando con claridad. Gracias por tomarte el tiempo de responder con honestidad y dar lo mejor de ti. Confío en tu criterio. ☀️

---

## ✿ PARTE 1 – CLASIFICACIÓN Y ANALOGÍA (30 puntos)

**Instrucciones:** Por cada patrón, marca:

- Su **clasificación** (C = Creacional, E = Estructural, B = Comportamiento)
- Su analogía correcta

**Valor:** 2 puntos por patrón (1 por clasificación + 1 por analogía) × 15 patrones = **30 puntos**

#	Patrón	Clasificación (C/E/B)		Descripción/Analogía
1	Singleton	<input type="radio"/> C	5	Clonar como copiar un archivo

<b>2</b>	Factory Method	<input type="radio"/> C	10	Cajero automático con botones simples
<b>3</b>	Builder	<input type="radio"/> C	12	Control remoto con botones programables
<b>4</b>	Abstract Factory	<input type="radio"/> C	6	Enchufe que adapta un conector
<b>5</b>	Prototype	<input type="radio"/> C	15	Guardar partida en un videojuego
<b>6</b>	Adapter	<input type="radio"/> E	11	Elegir método de pago
<b>7</b>	Bridge	<input type="radio"/> E	8	Añadir toppings a un helado
<b>8</b>	Decorator	<input type="radio"/> E	1	Solo puede existir una instancia global
<b>9</b>	Composite	<input type="radio"/> E	9	Carpetas y archivos organizados en árbol
<b>10</b>	Facade	<input type="radio"/> E	13	Suscripción a un canal de YouTube
<b>11</b>	Strategy	<input type="radio"/> B	14	Semáforo que cambia de color
<b>12</b>	Command	<input type="radio"/> B	4	Tienda que produce familias completas de objetos
<b>13</b>	Observer	<input type="radio"/> B	3	Armar una hamburguesa paso a paso
<b>14</b>	State	<input type="radio"/> B	2	Crea objetos sin saber qué clase concreta usar
<b>15</b>	Memento	<input type="radio"/> B	7	Control remoto desacoplado de la TV

---

## 🔍 PARTE 2 – CASOS DE USO (40 puntos)

**Instrucciones:** Elige el patrón que mejor se aplica a cada situación real.

**Valor:** 5 puntos por respuesta × 8 casos = **40 puntos**

1. Un juego de rol necesita guardar el progreso del jugador en diferentes niveles.

Memento

2. Una aplicación bancaria permite notificar automáticamente a varios módulos cuando se realiza un movimiento en una cuenta.

Observer

3. Un sistema necesita construir un objeto paso a paso, permitiendo distintas configuraciones como si se tratara de un combo de hamburguesas.

Builder

4. Un botón de una app ejecuta diferentes acciones según lo que el usuario haya elegido (abrir, guardar, imprimir).

Command

5. Una plataforma web debe acceder a recursos protegidos, pero necesita controlar los permisos antes de hacerlo.

Proxy

6. Una tienda online cambia el método de pago según el país del usuario (efectivo, tarjeta, transferencia).

Strategy

7. Un semáforo cambia su comportamiento según el color actual.

State

8. Un técnico de mantenimiento recorre varios dispositivos y debe aplicar una acción diferente en cada uno sin modificar las clases originales.

Visitor

- Escribe **solo el número del patrón** correcto. Si lo deseas, puedes justificar (opcional).
- 



## PARTE 3 – DESARROLLO PRÁCTICO (30 puntos)

### Instrucciones:

Elige **uno** de los patrones de diseño vistos en clase.

- Escribe un ejemplo de la vida real **diferente a los que vimos en clase**
- Implementa el ejemplo con un archivo .py (mínimo 3 clases, debe ejecutarse y mostrar su lógica)
- Puedes usar Google para inspirarte, pero **no uses inteligencia artificial generativa**



Se califica:

- Claridad (10 pts)
- Contenido funcional y coherente (10 pts)

- Creatividad y originalidad del ejemplo (10 pts)

Envía el archivo por Teams

Escogí la clase mediador, con las clases mediador, empleado, Desarrollador, Tester y LiderProyecto. La idea es crear una interacción entre ellos para centralizar la lógica de comunicación. Desarrollador, tester y líderProyecto pueden intercambiar mensajes entre si pero esto a través de un mediador que en este caso es el Coordinador.

class Mediador:

```
def enviar(self, mensaje, emisor):
    pass
```

class Empleado:

```
def __init__(self, nombre, mediador):
    self.nombre = nombre
    self.mediador = mediador
```

```
def enviar(self, mensaje):
    print(f"\n{self.nombre} dice: {mensaje}")
    self.mediador.enviar(mensaje, self)
```

```
def recibir(self, mensaje):
    print(f"{self.nombre} recibe: {mensaje}")
```

class CoordinadorProyecto(Mediador):

```
def __init__(self):
    self.empleados = []
```

```
def registrar(self, empleado):
```

```
    self.empleados.append(empleado)

def enviar(self, mensaje, emisor):
    for empleado in self.empleados:
        if empleado != emisor:
            empleado.recibir(f"{emisor.nombre}: {mensaje}")

class Desarrollador(Empleado):
    pass

class Tester(Empleado):
    pass

class LiderProyecto(Empleado):
    pass

def menu_oficina():
    coordinador = CoordinadorProyecto()
    dev = Desarrollador("Cris (Desarrollador)", coordinador)
    tester = Tester("Luis (Tester)", coordinador)
    lider = LiderProyecto("Danni (Lider)", coordinador)

    coordinador.registrar(dev)
    coordinador.registrar(tester)
    coordinador.registrar(lider)
```

```
empleados = {  
    "1": dev,  
    "2": tester,  
    "3": lider  
}  
  
while True:  
    print("\n-----Oficina Tres Patitos -----")  
    print("1. Cris (Desarrollador)")  
    print("2. Luis (Tester)")  
    print("3. Danni (Líder de Proyecto)")  
    print("4. Salir")  
    opcion = input("¿Quién quiere enviar un mensaje? ")  
  
    if opcion in empleados:  
        mensaje = input("Escribe tu mensaje: ")  
        empleados[opcion].enviar(mensaje)  
    elif opcion == "4":  
        print("Saliendo")  
        break  
    else:  
        print("Opción inválida, intenta de nuevo.")  
  
menu_oficina()
```

Escogí el patrón memento con la clase Tesis la cual tiene los métodos: escribir, mostrar, restaurar y salir. La idea es que el usuario pueda escribir una versión de su tesis y la guarde. Mientras siga escribiendo versiones estas se van guardando en espacios diferentes de memoria, si por alguna razón el usuario quiere regresar a una función específica, este puede hacerlo mediante la opción restaurar.

class Tesis:

```
def __init__(self):  
    self.contenido = ""  
  
def escribir(self, texto):  
    self.contenido += texto + "\n"
```

```
def crear_memento(self):  
    return Memento(self.contenido)
```

```
def restaurar(self, memento):  
    self.contenido = memento.estado
```

```
def mostrar(self):  
    print("\n Contenido actual de la tesis:\n" + self.contenido)
```

class Memento:

```
def __init__(self, estado):  
    self.estado = estado
```

```
def menu_tesis():  
    tesis = Tesis()
```

```

versiones = []

while True:

    print("\n==== TERMINA PORFIN DE TESIS ===")

    print("1. Escribir nueva versión")
    print("2. Mostrar versión actual")
    print("3. Restaurar versión anterior")
    print("4. Salir")

    opcion = input("Elige una opción: ")

    if opcion == "1":

        texto = input("Escribe el contenido de la nueva versión: ")

        tesis.escribir(texto)

        versiones.append(thesis.crear_memento())

        print(f" Versión {len(versiones)} guardada.")

    elif opcion == "2":

        thesis.mostrar()

    elif opcion == "3":

        if versiones:

            try:

                num = int(input(f"Ingrese número de versión a restaurar (1 - {len(versiones)}): "))

                if 1 <= num <= len(versiones):

                    thesis.restaurar(versiones[num - 1])

                    print(f"Restaurado a la versión {num}.")

            except ValueError:
                print("Por favor, ingrese un número válido.")

    else:
        break

```

```
    print("Número inválido.")

except ValueError:

    print("Ingresa un número válido.")

else:

    print("No hay versiones guardadas aún.")

elif opcion == "4":

    print("Saliendo")

    break

else:

    print("Opción no válida. Intenta otra vez.")

menu_tesis()
```