TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto - Diciembre 2025

CARRERA:
Ingeniería en Sistemas Computacionales

MATERIA:
Patrones de diseño

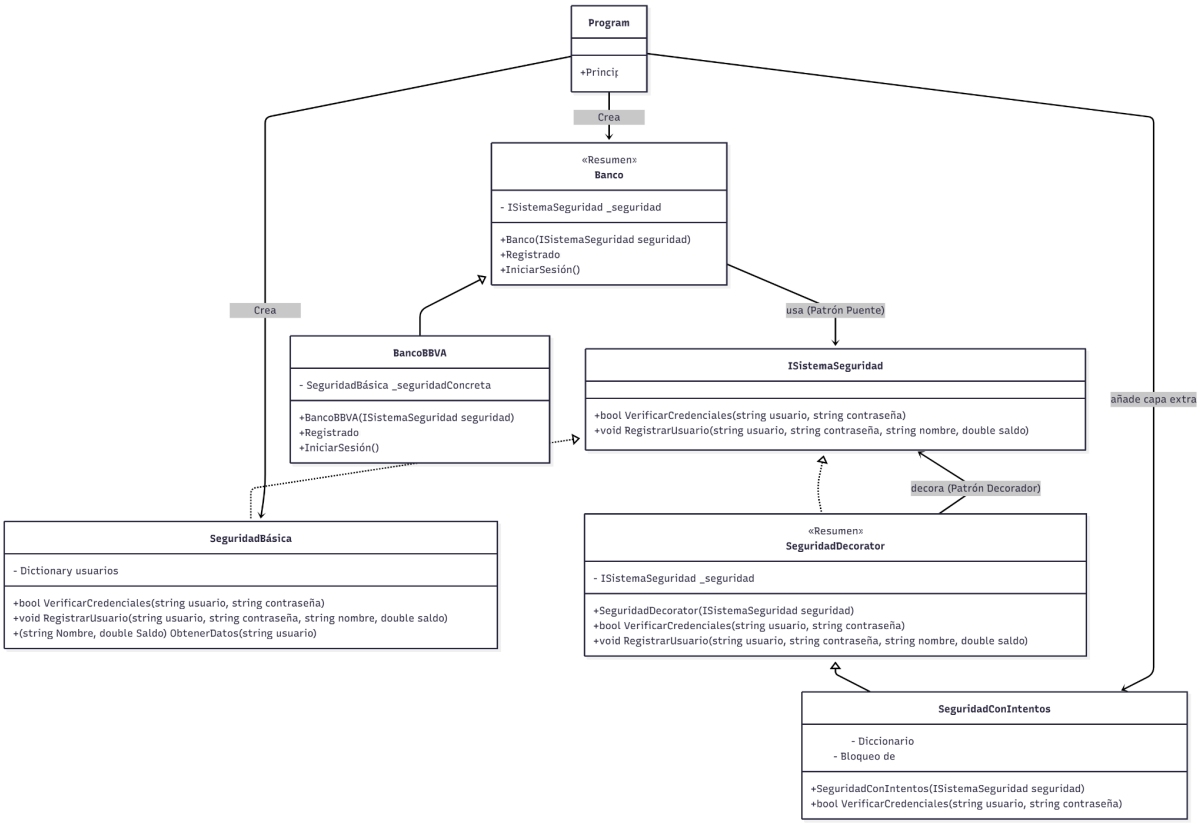
TÍTULO ACTIVIDAD:
Examen unidad 3

UNIDAD A EVALUAR:
Unidad 3

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Ceballos Resendiz Veronica Sarahi 20211759

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

DIAGRAMA UML



```
CODIGOusing System;
using System.Collections.Generic;
using System.Threading;

// ================================================
// SISTEMA DE AUTENTICACIÓN BANCO BBVA
// Usa los patrones de diseño: DECORADOR + PUENTE
// ================================================

// ---------- PATRÓN PUENTE ----------
// Interfaz que define la implementación del sistema de seguridad.
public interface ISistemaSeguridad
{
    bool VerificarCredenciales(string usuario, string contraseña);
    void RegistrarUsuario(string usuario, string contraseña, string nombre, double
saldo);
}

// ---------- PATRÓN PUENTE ----------
// Implementación concreta del sistema de seguridad.
public class SeguridadBasica : ISistemaSeguridad
{
    private Dictionary<string, (string Contraseña, string Nombre, double Saldo)>
usuarios =
        new Dictionary<string, (string, string, double)>();

    public virtual bool VerificarCredenciales(string usuario, string contraseña)
    {
```

```
        if (usuarios.ContainsKey(usuario) && usuarios[usuario].Contraseña ==
contraseña)
        {
            return true;
        }
        return false;
    }

    public virtual void RegistrarUsuario(string usuario, string contraseña, string
nombre, double saldo)
    {
        if (!usuarios.ContainsKey(usuario))
        {
            usuarios.Add(usuario, (contraseña, nombre, saldo));
            Console.WriteLine("Registro exitoso.");
        }
        else
        {
            Console.WriteLine("El usuario ya existe.");
        }
    }

    public (string Nombre, double Saldo) ObtenerDatos(string usuario)
    {
        if (usuarios.ContainsKey(usuario))
            return (usuarios[usuario].Nombre, usuarios[usuario].Saldo);
        return ("Desconocido", 0);
    }
```

```
}

// ---------- PATRÓN DECORADOR ----------
// Clase abstracta que añade nuevas capas de seguridad sin modificar la clase base.
public abstract class SeguridadDecorator : ISistemaSeguridad
{
    protected ISistemaSeguridad _seguridad;

    public SeguridadDecorator(ISistemaSeguridad seguridad)
    {
        _seguridad = seguridad;
    }

    public virtual bool VerificarCredenciales(string usuario, string contraseña)
    {
        return _seguridad.VerificarCredenciales(usuario, contraseña);
    }

    public virtual void RegistrarUsuario(string usuario, string contraseña, string
nombre, double saldo)
    {
        _seguridad.RegistrarUsuario(usuario, contraseña, nombre, saldo);
    }
}

// ---------- PATRÓN DECORADOR ----------
// Decorador concreto que agrega intentos fallidos y bloqueos temporales o
permanentes.
```

```
public class SeguridadConIntentos : SeguridadDecorator
{
    private Dictionary<string, int> intentosFallidos = new Dictionary<string, int>();
    private Dictionary<string, DateTime> bloqueos = new Dictionary<string,
DateTime>();

    public SeguridadConIntentos(ISistemaSeguridad seguridad) : base(seguridad) { }

    public override bool VerificarCredenciales(string usuario, string contraseña)
    {
        // Verifica si la cuenta está bloqueada temporalmente o por un día.
        if (bloqueos.ContainsKey(usuario))
        {
            TimeSpan diferencia = DateTime.Now - bloqueos[usuario];

            if (diferencia.TotalMinutes < 3)
            {
                Console.WriteLine($"Cuenta bloqueada. Espera {Math.Ceiling(3 -
diferencia.TotalMinutes)} minuto(s).");
                return false;
            }
            else if (diferencia.TotalDays < 1 && diferencia.TotalMinutes >= 3)
            {
                Console.WriteLine("Tu cuenta está bloqueada por seguridad durante 1
día.");
                Console.WriteLine("Comunícate a soporte BBVA: 55-5226-2663 o acude a
tu sucursal más cercana.");
                return false;
```

```
            }
            else
            {
                bloqueos.Remove(usuario);
                intentosFallidos[usuario] = 0;
            }
        }

        bool valido = _seguridad.VerificarCredenciales(usuario, contraseña);

        if (valido)
        {
            intentosFallidos[usuario] = 0;
            return true;
        }
        else
        {
            if (!intentosFallidos.ContainsKey(usuario)) intentosFallidos[usuario] = 0;
            intentosFallidos[usuario]++;

            if (intentosFallidos[usuario] == 3)
            {
                Console.WriteLine("Has fallado 3 veces. Espera 3 minutos antes de volver
a intentar.");
                bloqueos[usuario] = DateTime.Now;
            }
            else if (intentosFallidos[usuario] > 3)
```

```
            {
                Console.WriteLine("Tu cuenta ha sido bloqueada por seguridad durante 1 día.");
                Console.WriteLine("Comunícate a soporte BBVA: 55-5226-2663 o acude a tu sucursal más cercana.");
                bloqueos[usuario] = DateTime.Now.AddDays(1);
            }
            else
            {
                Console.WriteLine($"Contraseña incorrecta. Intentos restantes: {3 - intentosFallidos[usuario]}");
            }

            return false;
        }
    }
}

// ---------- PATRÓN PUENTE ----------
// Clase abstracta que conecta la parte lógica (Banco) con el sistema de seguridad.
public abstract class Banco
{
    protected ISistemaSeguridad _seguridad;

    public Banco(ISistemaSeguridad seguridad)
    {
        _seguridad = seguridad;
    }
```

```
    public abstract void Registrar();
    public abstract void IniciarSesion();
}

// ---------- IMPLEMENTACIÓN CONCRETA DEL PUENTE ----------
public class BancoBBVA : Banco
{
    private SeguridadBasica _seguridadConcreta;

    public BancoBBVA(ISistemaSeguridad seguridad) : base(seguridad)
    {
        if (seguridad is SeguridadDecorator dec)
        {
            var campo = typeof(SeguridadDecorator).GetField("_seguridad",
                System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance);

            _seguridadConcreta = (SeguridadBasica)(campo?.GetValue(dec) ??
seguridad);
        }
        else
        {
            _seguridadConcreta = (SeguridadBasica)seguridad;
        }
    }

    public override void Registrar()
```

```
{
    Console.Clear();
    Console.WriteLine("=== Registro BBVA ===");
    Console.Write("Nombre completo: ");
    string nombre = Console.ReadLine();
    Console.Write("Usuario: ");
    string usuario = Console.ReadLine();
    Console.Write("Contraseña: ");
    string contraseña = Console.ReadLine();
    Console.Write("Saldo inicial: $");
    double saldo = Convert.ToDouble(Console.ReadLine());

    _seguridad.RegistrarUsuario(usuario, contraseña, nombre, saldo);
    Console.WriteLine("\nPresiona una tecla para volver al menú...");
    Console.ReadKey();
}

public override void IniciarSesion()
{
    Console.Clear();
    Console.WriteLine("=== Iniciar Sesión BBVA ===");
    Console.Write("Usuario: ");
    string usuario = Console.ReadLine();

    int intentos = 0;
    bool acceso = false;
```

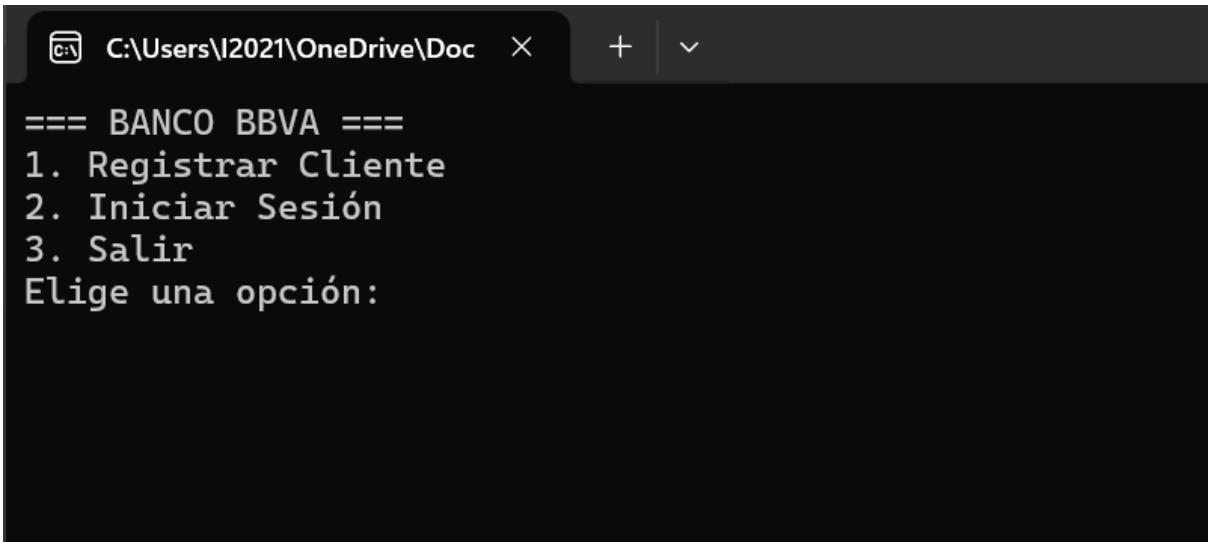
```
while (intentos < 4 && !acceso)
{
    Console.Write("Contraseña: ");
    string contraseña = "";
    ConsoleKeyInfo tecla;

    // Captura de contraseña oculta (sin mostrar texto)
    do
    {
        tecla = Console.ReadKey(true);
        if (tecla.Key != ConsoleKey.Backspace && tecla.Key != ConsoleKey.Enter)
        {
            contraseña += tecla.KeyChar;
            Console.Write("*");
        }
        else if (tecla.Key == ConsoleKey.Backspace && contraseña.Length > 0)
        {
            contraseña = contraseña.Substring(0, contraseña.Length - 1);
            Console.Write("\b \b");
        }
    } while (tecla.Key != ConsoleKey.Enter);

    Console.WriteLine();

    if (_seguridad.VerificarCredenciales(usuario, contraseña))
    {
        acceso = true;
```

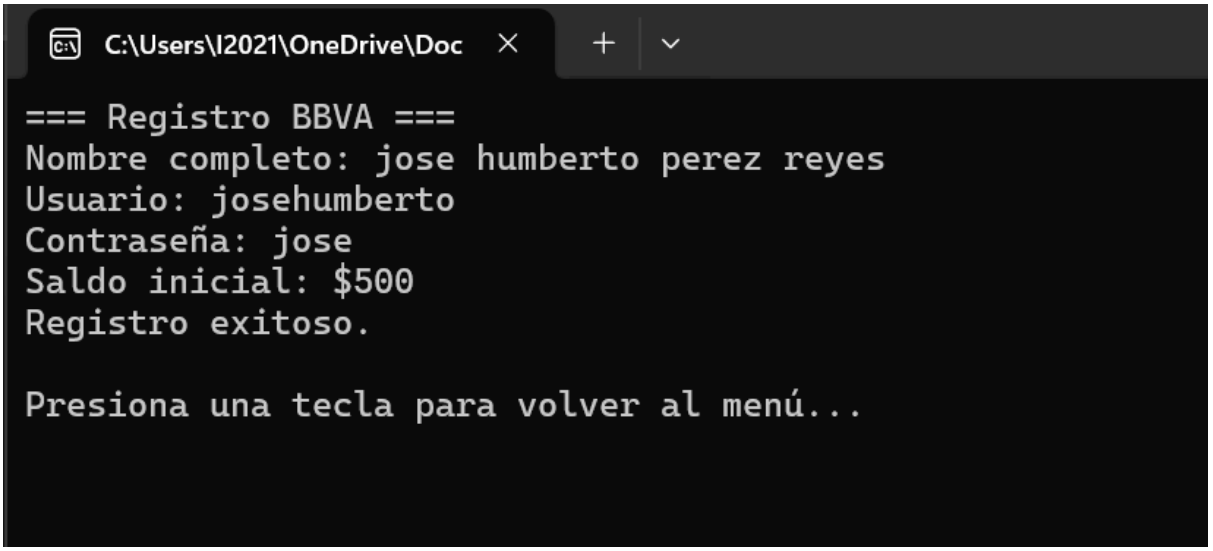
```
            Console.Clear();
            var datos = _seguridadConcreta.ObtenerDatos(usuario);
            Console.WriteLine("Inicio de sesión exitoso.");
            Console.WriteLine($"\nNombre: {datos.Nombre}");
            Console.WriteLine($"Saldo disponible: ${datos.Saldo}");
            Console.WriteLine("\nPresiona una tecla para volver al menú...");
            Console.ReadKey();
        }
        else
        {
            intentos++;
            if (intentos < 4)
            {
                Console.WriteLine("\nPresiona una tecla para intentar nuevamente...");
                Console.ReadKey();
                Console.Clear();
                Console.WriteLine("=== Iniciar Sesión BBVA ===");
                Console.WriteLine($"Intento {intentos + 1} de 4");
                Console.Write($"Usuario: {usuario}\n");
            }
            else
            {
                Console.WriteLine("\nSe acabaron tus intentos. Regresando al menú
principal...");
                Thread.Sleep(2000);
            }
        }
```

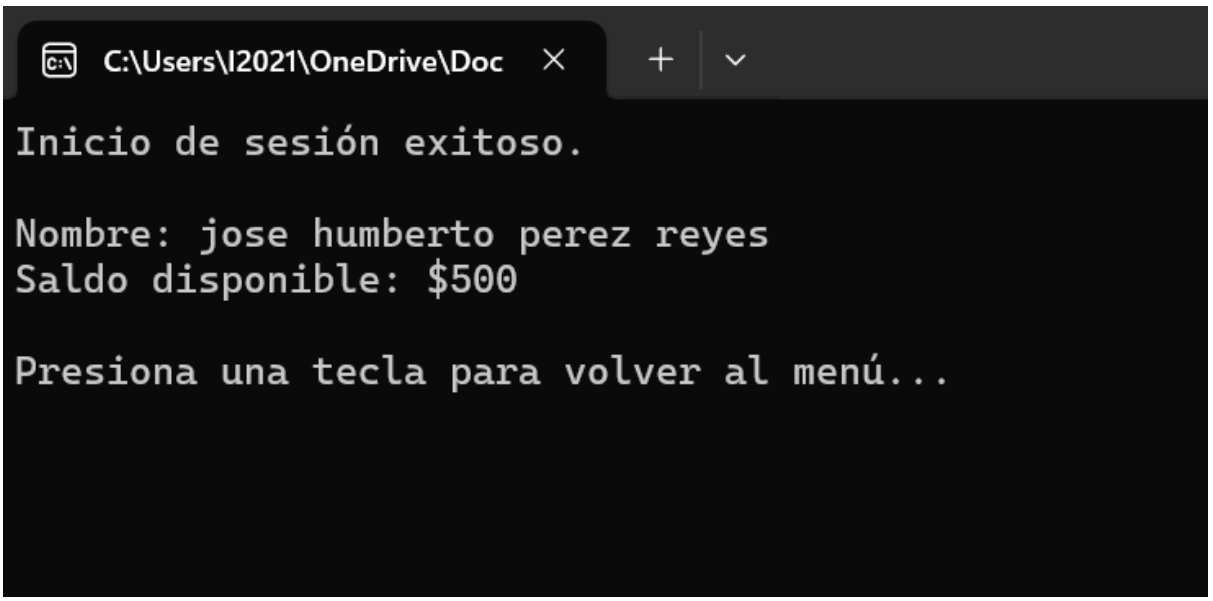
```
        }
    }
}

// ---------- CLASE PRINCIPAL (CLIENTE) ----------
public class Program
{
    public static void Main()
    {
        // PATRÓN DECORADOR: Añade control de intentos y bloqueos al sistema de seguridad.
        ISistemaSeguridad seguridad = new SeguridadConIntentos(new SeguridadBasica());

        // PATRÓN PUENTE: Conecta la abstracción Banco con la implementación de seguridad.
        Banco banco = new BancoBBVA(seguridad);

        int opcion;
        do
        {
            Console.Clear();
            Console.WriteLine("=== BANCO BBVA ===");
            Console.WriteLine("1. Registrar Cliente");
            Console.WriteLine("2. Iniciar Sesión");
            Console.WriteLine("3. Salir");
            Console.Write("Elige una opción: ");
            if (!int.TryParse(Console.ReadLine(), out opcion)) opcion = 0;
```

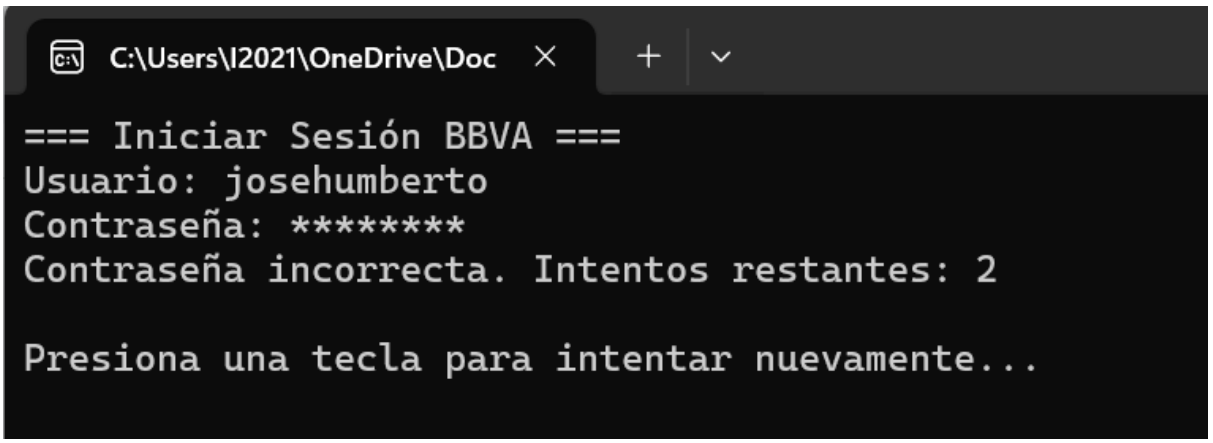
```
            switch (opcion)
            {
                case 1:
                    banco.Registrar();
                    break;
                case 2:
                    banco.IniciarSesion();
                    break;
                case 3:
                    Console.WriteLine("Gracias por usar BBVA. Hasta pronto.");
                    break;
                default:
                    Console.WriteLine("Opción no válida.");
                    Thread.Sleep(1500);
                    break;
            }
        } while (opcion != 3);
    }
}
```

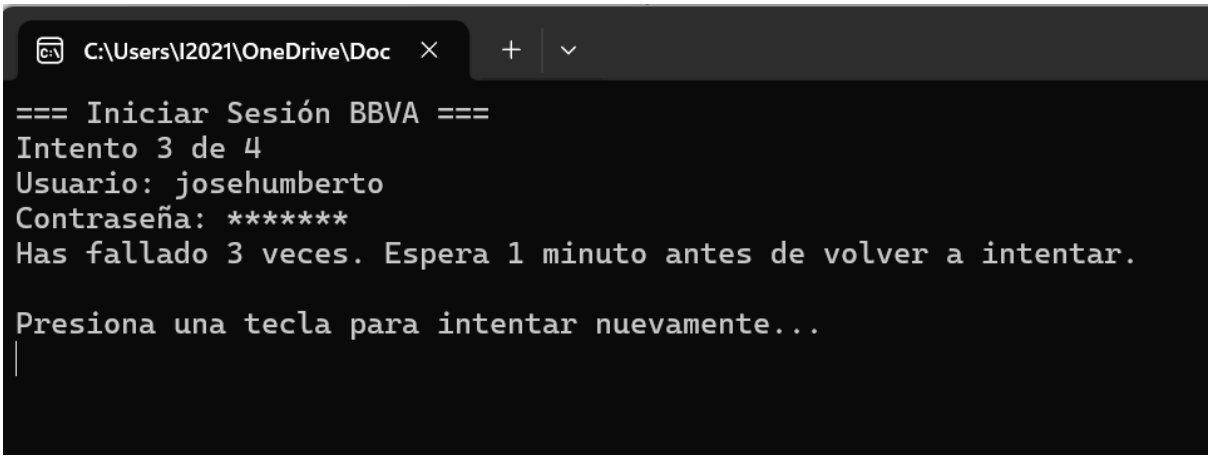
CAPTURAS


C:\Users\I2021\OneDrive\Doc
=== BANCO BBVA ===
1. Registrar Cliente
2. Iniciar Sesión
3. Salir
Elige una opción:


C:\Users\I2021\OneDrive\Doc
=== Registro BBVA ===
Nombre completo: jose humberto perez reyes
Usuario: josehumberto
Contraseña: jose
Saldo inicial: $500
Registro exitoso.
Presiona una tecla para volver al menú...


C:\Users\I2021\OneDrive\Doc
Inicio de sesión exitoso.
Nombre: jose humberto perez reyes
Saldo disponible: $500
Presiona una tecla para volver al menú...





Conclusión

Lo que hace el programa es la autenticación en capas lo que hace el patrón decorador es que en este caso es la seguridad del programa es que se pueden agregar varios métodos de seguridad sin necesidad de dañar el programa completo o sin necesidad de que modificar el programa, solo se le añadirían más métodos de seguridad. El puente actúa casi de la misma manera pero con la clase abstracta de banco y gracias a que se implementa este patrón se pueden crear muchas más abstracciones y tipos de sistemas de seguridad sin que dependan unas de otras así cada banco puede tener su tipo sin tener que alterar el código.