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

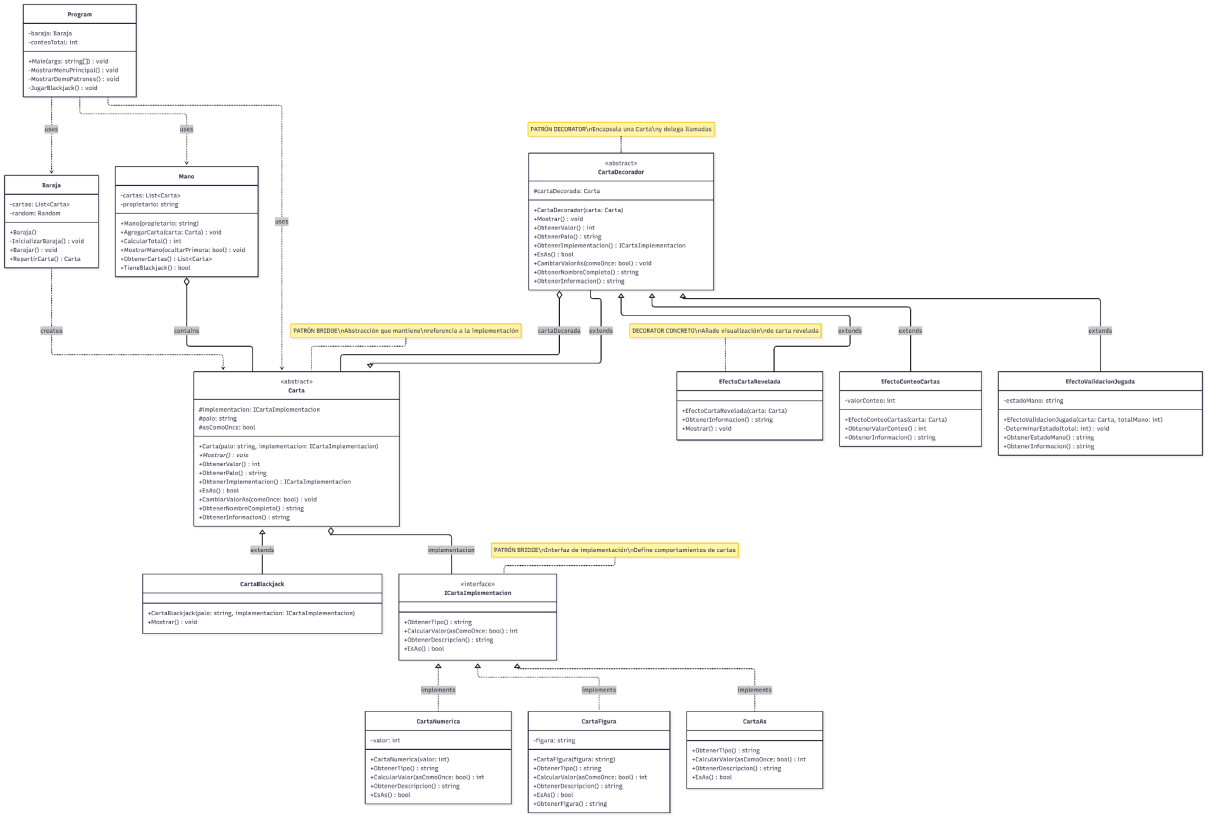
UNIDAD 3

NOMBRE Y NÚMERO DE CONTROL:
Ochoa Moran Victor Alejandro 22210329

NOMBRE DEL MAESTRO (A):

MARIBEL GUERRERO LUIS

Diagrama UML



Código

Patron Bridge

Abstracción

Carta.cs

```
public abstract class Carta

{

    protected ICartaImplementacion implementacion;

    protected string palo;

    protected bool asComoOnce;



    public Carta(string palo, ICartaImplementacion implementacion)

    {

        this.palo = palo;

        this.implementacion = implementacion;

        this.asComoOnce = true;

    }



    public abstract void Mostrar();



    public virtual int ObtenerValor()

    {

        return implementacion.CalcularValor(asComoOnce);

    }



    public virtual string ObtenerPalo()

    {

        return palo;
```

```
    }

    public virtual ICartaImplementacion ObtenerImplementacion()
    {
        return implementacion;
    }

    public virtual bool EsAs()
    {
        return implementacion.EsAs();
    }

    public virtual void CambiarValorAs(bool comoOnce)
    {
        if (EsAs())
        {
            asComoOnce = comoOnce;
        }
    }

    public virtual string ObtenerNombreCompleto()
    {
        string nombre = "";

        if (implementacion is CartaNumerica)
        {
            nombre = implementacion.CalcularValor(true).ToString();
```

```
        }
        else if (implementacion is CartaFigura)
        {
            int valor = implementacion.CalcularValor(true);
            if (valor == 10)
            {
                nombre = implementacion.ObtenerTipo() == "Figura" ?
"J/Q/K" : "10";

                string desc = implementacion.ObtenerDescripcion();
                if (desc.Contains("Jota")) nombre = "J";
                else if (desc.Contains("Reina")) nombre = "Q";
                else if (desc.Contains("Rey")) nombre = "K";
            }
        }
        else if (implementacion is CartaAs)
        {
            nombre = "A";
        }

        return $"{nombre} de {palo}";
    }

    public virtual string ObtenerInformacion()
    {
        return $"Carta: {ObtenerNombreCompleto()}\n" +
               $"Tipo: {implementacion.ObtenerTipo()}\n" +
```

```
                $"Valor actual: {ObtenerValor()} puntos\n" +
                $"Descripción: {implementacion.ObtenerDescripcion()}";
    }
}
```

CrataBlackJack.cs

```
public class CartaBlackjack : Carta
{
    public CartaBlackjack(string palo, ICartaImplementacion
implementacion)
        : base(palo, implementacion)
    {
    }

    public override void Mostrar()
    {
        Console.WriteLine(ObtenerInformacion());
    }
}
```

Implementación

Carta As.cs

```
public class CartaAs : ICartaImplementacion

{

    public string ObtenerTipo()

    {

        return "As";

    }



    public int CalcularValor(bool asComoOnce)

    {



        if (asComoOnce == true)

        {

            return 11;

        }

        else

        {

            return 1;

        }



    }



    public string ObtenerDescripcion()

    {

        return "As - Puede valer 1 u 11 puntos según convenga";

    }
```

```
    public bool EsAs()
    {
        return true;
    }
}
```

CartaFiguara.cs

```
public class CartaFigura : ICartaImplementacion
{
    private string figura;

    public CartaFigura(string figura)
    {
        if (figura != "J" && figura != "Q" && figura != "K")
            throw new ArgumentException("La figura debe ser J, Q o K");

        this.figura = figura;
    }

    public string ObtenerTipo()
    {
        return "Figura";
    }

    public int CalcularValor(bool asComoOnce)
    {
```

```
        return 10;
    }

    public string ObtenerDescripcion()
    {
        string nombreFigura;

        if (figura == "J")
        {
            nombreFigura = "Jota";
        }
        else
        {
        if (figura == "Q")
        {
            nombreFigura = "Reina";
        }
        else
        {
        nombreFigura = "Rey";
        }
    }
        return ${nombreFigura} - Vale 10 puntos";
}

    public bool EsAs()
    {
```

```
        return false;
    }



}
```

CartaNumerica.cs

```
public class CartaNumerica : ICartaImplementacion
{
    private int valor;

    public CartaNumerica(int valor)
    {
        if (valor < 2 || valor > 10)
            throw new ArgumentException("Las cartas numéricas deben
tener valor entre 2 y 10");

        this.valor = valor;
    }

    public string ObtenerTipo()
    {
        return "Numérica";
    }

    public int CalcularValor(bool asComoOnce)
    {
```

```
        return valor;
    }

    public string ObtenerDescripcion()
    {
        return $"Carta numérica con valor fijo de {valor}";
    }

    public bool EsAs()
    {
        return false;
    }
}
```

ICartaImplementacion.cs

```
public interface ICartaImplementacion
{
    string ObtenerTipo();
    int CalcularValor(bool asComoOnce);
    string ObtenerDescripcion();
    bool EsAs();
}
```

Patrón Decorador

Cartadecorador.cs

```
public abstract class CartaDecorador : Carta

{

    protected Carta cartaDecorada;



    public CartaDecorador(Carta carta)

        : base(carta.ObtenerPalo(), carta.ObtenerImplementacion())

    {

        this.cartaDecorada = carta;

    }



    public override void Mostrar()

    {

        cartaDecorada.Mostrar();

    }



    public override int ObtenerValor()

    {

        return cartaDecorada.ObtenerValor();

    }



    public override string ObtenerPalo()

    {

        return cartaDecorada.ObtenerPalo();

    }
```

```
    public override ICartaImplementacion ObtenerImplementacion()
    {
        return cartaDecorada.ObtenerImplementacion();
    }

    public override bool EsAs()
    {
        return cartaDecorada.EsAs();
    }

    public override void CambiarValorAs(bool comoOnce)
    {
        cartaDecorada.CambiarValorAs(comoOnce);
    }

    public override string ObtenerNombreCompleto()
    {
        return cartaDecorada.ObtenerNombreCompleto();
    }

    public override string ObtenerInformacion()
    {
        return cartaDecorada.ObtenerInformacion();
    }
}
```

Efecto Carta elevada.cs

```
public class EfectoCartaRevelada : CartaDecorador

{

    public EfectoCartaRevelada(Carta carta) : base(carta)

    {

    }



    public override string ObtenerInformacion()

    {

        return cartaDecorada.ObtenerInformacion() +

               "\n[CARTA REVELADA] ✓ Visible para todos los
jugadores";

    }



    public override void Mostrar()

    {

        Console.WriteLine("┌──────────────────────────────────────┐");

        base.Mostrar();

        Console.WriteLine("└──────────────────────────────────────┘");

    }

}
```

Efecto Conteo Cartas.cs

```
public class EfectoCartaRevelada : CartaDecorador

{

    public EfectoCartaRevelada(Carta carta) : base(carta)

    {

    }



    public override string ObtenerInformacion()

    {

        return cartaDecorada.ObtenerInformacion() +

               "\n[CARTA REVELADA] ✓ Visible para todos los
jugadores";

    }



    public override void Mostrar()

    {

        Console.WriteLine("┌──────────────────────────────────────┐");

        base.Mostrar();

        Console.WriteLine("└──────────────────────────────────────┘");

    }

}
```

Efecto validacion jugada.cs

```
public class EfectoValidacionJugada : CartaDecorador

{

    private string estadoMano;



    public EfectoValidacionJugada(Carta carta, int totalMano) :
base(carta)

    {

        DeterminarEstado(totalMano);

    }



    private void DeterminarEstado(int total)

    {

        if (total == 21)

            estadoMano = "¡BLACKJACK! ";

        else if (total > 21)

            estadoMano = "¡TE PASASTE! ";

        else if (total >= 17)

            estadoMano = "Mano fuerte ";

        else if (total >= 12)

            estadoMano = "Mano media ";

        else

            estadoMano = "Mano débil ";

    }



    public string ObtenerEstadoMano()

    {
```

```
        return estadoMano;
    }

    public override string ObtenerInformacion()
    {
        return cartaDecorada.ObtenerInformacion() +
               $"\n[VALIDACIÓN] Estado de la mano: {estadoMano}";
    }
}
```

Program.cs

```
class Program

{

    static Baraja baraja;

    static int conteoTotal = 0;



    static void Main(string[] args)

    {

        Console.OutputEncoding = System.Text.Encoding.UTF8;



        bool jugarOtraVez = true;



        while (jugarOtraVez)

        {

            MostrarMenuPrincipal();

            string opcion = Console.ReadLine();



            switch (opcion)

            {

                case "1":

                    JugarBlackjack();

                    break;

                case "2":

                    Console.WriteLine("\n ¡Gracias por jugar! ¡Hasta
pronto!");

                    jugarOtraVez = false;

                    break;
```

```
                default:
                    Console.WriteLine("\n Opción inválida. Intenta de
nuevo.");
                    break;
            }
        }
    }

    static void MostrarMenuPrincipal()
    {
        Console.Clear();

Console.WriteLine("╔═══════════════════════════════════════════════════
═══════╗");
        Console.WriteLine("║      BLACKJACK - PATRONES BRIDGE Y
DECORATOR           ║");

Console.WriteLine("╚═══════════════════════════════════════════════════
═══════╝\n");
        Console.WriteLine("Selecciona una opción:\n");
        Console.WriteLine("  1)  Jugar Blackjack");
        Console.WriteLine("  2)  Salir\n");
        Console.Write(" Tu elección: ");
    }


    static void JugarBlackjack()
    {
        Console.Clear();
```

```
Console.WriteLine("╔═══════════════════════════════════════════════════════════════════════════════╗");
        Console.WriteLine("║                    PARTIDA DE BLACKJACK                                       ║");
Console.WriteLine("╚═══════════════════════════════════════════════════════════════════════════════╝\n");

        baraja = new Baraja();
        baraja.Barajar();
        conteoTotal = 0;

        Mano manoJugador = new Mano("JUGADOR");
        Mano manoCrupier = new Mano("CRUPIER");

        Console.WriteLine("🎴 Repartiendo cartas...\n");
        System.Threading.Thread.Sleep(1000);

        manoJugador.AgregarCarta(baraja.RepartirCarta());
        manoCrupier.AgregarCarta(baraja.RepartirCarta());
        manoJugador.AgregarCarta(baraja.RepartirCarta());
        manoCrupier.AgregarCarta(baraja.RepartirCarta());

        manoJugador.MostrarMano();
        manoCrupier.MostrarMano(true);

        if (manoJugador.TieneBlackjack())
        {
```

```
            Console.WriteLine("\n ¡BLACKJACK! ");
            manoCrupier.MostrarMano();

            if (manoCrupier.TieneBlackjack())
            {
                Console.WriteLine("\n ¡EMPATE! El crupier también tiene
Blackjack.");
            }
            else
            {
                Console.WriteLine("\n ¡GANASTE! Blackjack natural.");
            }

            Console.WriteLine("\n Presiona cualquier tecla para volver
al menú...");
            Console.ReadKey();
            return;
        }


        bool sePlanto = false;
        while (!sePlanto && manoJugador.CalcularTotal() < 21)
        {
            Console.WriteLine("\n" + new string('—', 55));
            Console.WriteLine("¿Qué deseas hacer?");
            Console.WriteLine("  [C] Pedir Carta");
            Console.WriteLine("  [P] Plantarse");
            Console.Write("\n  Tu elección: ");
```

```
            string eleccion = Console.ReadLine().ToUpper();

            if (eleccion == "C")
            {
                Console.WriteLine("\n🎴 Pidiendo carta...\n");
                System.Threading.Thread.Sleep(800);

                Carta nuevaCarta = baraja.RepartirCarta();
                manoJugador.AgregarCarta(nuevaCarta);

                manoJugador.MostrarMano();

                if (manoJugador.CalcularTotal() > 21)
                {
                    Console.WriteLine("\n ¡TE PASASTE DE 21! Has perdido.");
                    Console.WriteLine("\n Presiona cualquier tecla para volver al menú...");
                    Console.ReadKey();
                    return;
                }
            }
            else if (eleccion == "P")
            {
                sePlanto = true;
                Console.WriteLine("\n Te has plantado con " + manoJugador.CalcularTotal() + " puntos.");
```

```
            }
            else
            {
                Console.WriteLine("\n Opción inválida. Usa C o P.");
            }
        }



        Console.WriteLine("\n" + new string('=', 55));
        Console.WriteLine(" Turno del crupier...\n");
        System.Threading.Thread.Sleep(1500);

        manoCrupier.MostrarMano();

        while (manoCrupier.CalcularTotal() < 17)
        {
            Console.WriteLine("\n🎴 El crupier pide carta...\n");
            System.Threading.Thread.Sleep(1500);

            manoCrupier.AgregarCarta(baraja.RepartirCarta());
            manoCrupier.MostrarMano();
        }



        int totalJugador = manoJugador.CalcularTotal();
        int totalCrupier = manoCrupier.CalcularTotal();
```

```
        Console.WriteLine("\n" + new string('=', 55));
        Console.WriteLine("              RESULTADO FINAL ");
        Console.WriteLine(new string('=', 55));
        Console.WriteLine($"  Jugador: {totalJugador} puntos");
        Console.WriteLine($"  Crupier: {totalCrupier} puntos");
        Console.WriteLine(new string('=', 55) + "\n");

        if (totalCrupier > 21)
        {
            Console.WriteLine(" ¡GANASTE! El crupier se pasó de 21.");
        }
        else if (totalJugador > totalCrupier)
        {
            Console.WriteLine(" ¡GANASTE! Tu mano es mejor.");
        }
        else if (totalJugador < totalCrupier)
        {
            Console.WriteLine(" Perdiste. El crupier tiene mejor
mano.");
        }
        else
        {
            Console.WriteLine(" ¡EMPATE! Misma puntuación.");
        }

        Console.WriteLine("\n Presiona cualquier tecla para volver al
menú...");
        Console.ReadKey();
```

```
    }
}
```

Baraja.cs

```
public class Baraja
{
    private List<Carta> cartas;
    private Random random;

    public Baraja()
    {
        cartas = new List<Carta>();
        random = new Random();
        InicializarBaraja();
    }

    private void InicializarBaraja()
    {
        string[] palos = { "Corazones", "Diamantes", "Tréboles",
"Picas" };

        foreach (string palo in palos)
        {

            for (int i = 2; i <= 10; i++)
            {
                cartas.Add(new CartaBlackjack(palo, new
CartaNumerica(i)));
```

```
            }



            cartas.Add(new CartaBlackjack(palo, new CartaFigura("J")));
            cartas.Add(new CartaBlackjack(palo, new CartaFigura("Q")));
            cartas.Add(new CartaBlackjack(palo, new CartaFigura("K")));



            cartas.Add(new CartaBlackjack(palo, new CartaAs()));
        }
    }

    public void Barajar()
    {
        for (int i = cartas.Count - 1; i > 0; i--)
        {
            int j = random.Next(i + 1);
            var temp = cartas[i];
            cartas[i] = cartas[j];
            cartas[j] = temp;
        }
    }

    public Carta RepartirCarta()
    {
        if (cartas.Count == 0)
        {
```

```
            Console.WriteLine(" La baraja se ha agotado,
reiniciando...");
            InicializarBaraja();
            Barajar();
        }

        Carta carta = cartas[0];
        cartas.RemoveAt(0);
        return new EfectoCartaRevelada(carta);
    }
}
```

Mano.cs

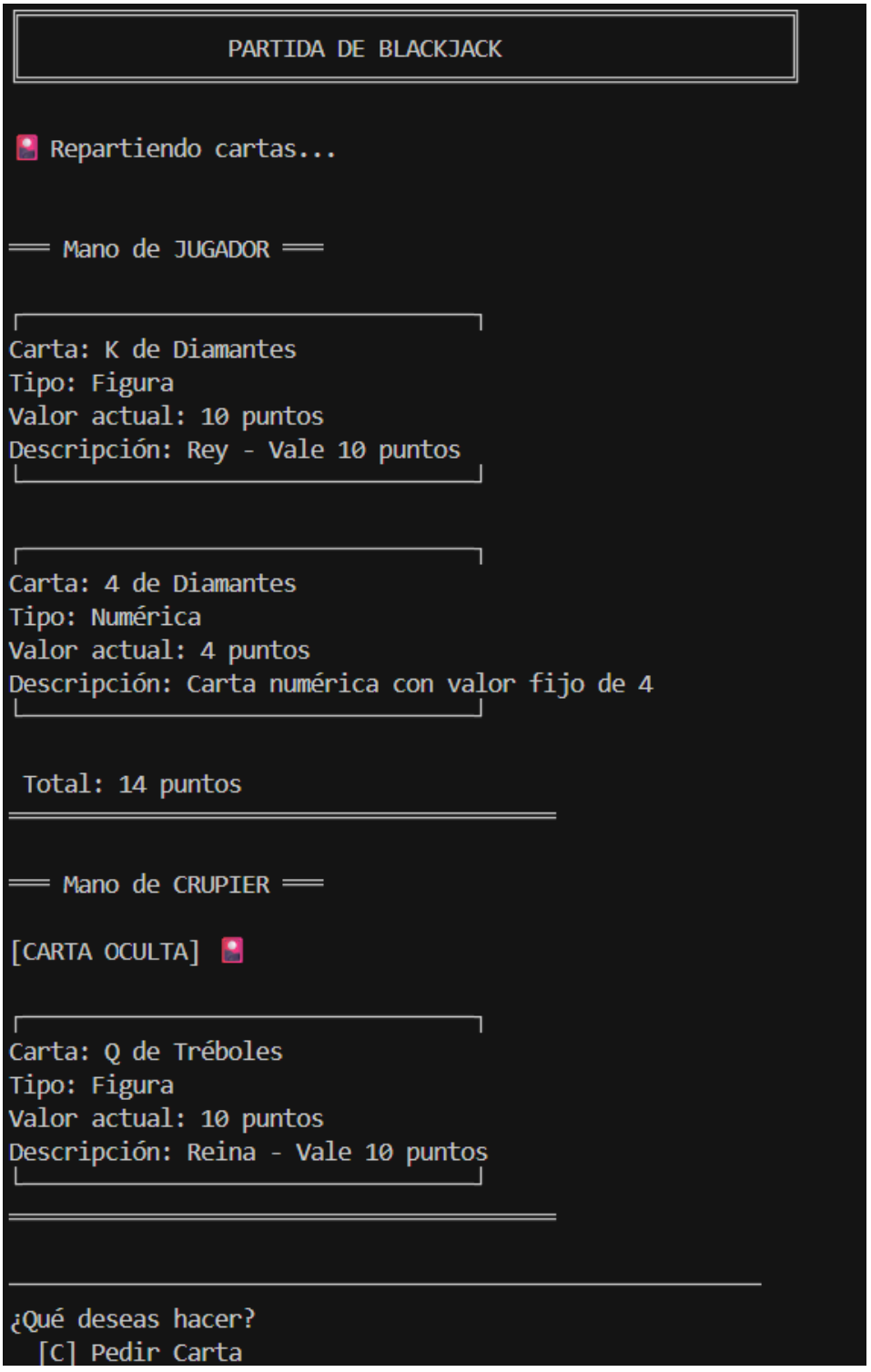
```
public class Mano
{
    private List<Carta> cartas;
    private string propietario;

    public Mano(string propietario)
    {
        this.propietario = propietario;
        this.cartas = new List<Carta>();
    }

    public void AgregarCarta(Carta carta)
    {
        cartas.Add(carta);
```

```
    }

    public int CalcularTotal()
    {
        int total = 0;
        int ases = 0;

        foreach (var carta in cartas)
        {
            total += carta.ObtenerValor();
            if (carta.EsAs())
                ases++;
        }

        while (total > 21 && ases > 0)
        {
            total -= 10;
            ases--;
        }

        return total;
    }

    public void MostrarMano(bool ocultarPrimera = false)
    {
        Console.WriteLine($"\n=== Mano de {propietario} ===");
```

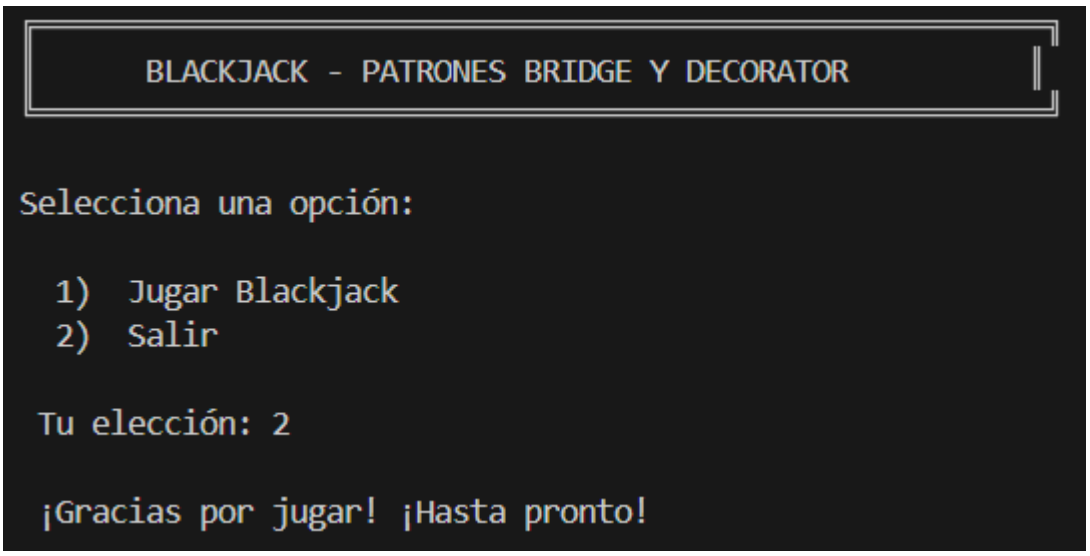
```
        for (int i = 0; i < cartas.Count; i++)
        {
            if (i == 0 && ocultarPrimera)
            {
                Console.WriteLine("\n[CARTA OCULTA] 🎴");
            }
            else
            {
                Console.WriteLine();
                cartas[i].Mostrar();
            }
        }

        if (!ocultarPrimera)
        {
            Console.WriteLine($"\n Total: {CalcularTotal()} puntos");
        }

        Console.WriteLine(new string('=', 40));
    }

    public List<Carta> ObtenerCartas()
    {
        return cartas;
    }

    public bool TieneBlackjack()
```

```
    {
        return cartas.Count == 2 && CalcularTotal() == 21;
    }
}
```

Captura del programa corriendo



```
  [P] Plantarse

  Tu elección: p

 Te has plantado con 14 puntos.

═══════════════════════════════════════════════════════════════

 Turno del crupier...


═══ Mano de CRUPIER ═══

┌──────────────────────────────────────┐
Carta: J de Tréboles
Tipo: Figura
Valor actual: 10 puntos
Descripción: Jota - Vale 10 puntos
└──────────────────────────────────────┘

┌──────────────────────────────────────┐
Carta: Q de Tréboles
Tipo: Figura
Valor actual: 10 puntos
Descripción: Reina - Vale 10 puntos
└──────────────────────────────────────┘

 Total: 20 puntos
═══════════════════════════════════════

═══════════════════════════════════════════════════════════════
                RESULTADO FINAL
═══════════════════════════════════════════════════════════════
  Jugador: 14 puntos
  Crupier: 20 puntos
═══════════════════════════════════════════════════════════════

 Perdiste. El crupier tiene mejor mano.

 Presiona cualquier tecla para volver al menú...
```


BLACKJACK - PATRONES BRIDGE Y DECORATOR
Selecciona una opción:
1) Jugar Blackjack
2) Salir
Tu elección: 2
¡Gracias por jugar! ¡Hasta pronto!

Conclusión

La implementación de los patrones “Bridge” y “Decorator” en este programa hace presente que es una combinación interesante, ya que el uso de patrón Bridge separa la abstracción de las cartas de su implementación concreta esto permite que ambas jerarquías evolucionen de manera independiente, lo que se demuestra en la “Carta” puede combinarse con cualquier tipo de implementación sin modificar la estructura base, facilitando la extensibilidad del sistema.

Por otro lado el patrón “Decorator” añade funcionalidades dinámicas a las cartas sin alterar su estructura original, mediante la encapsulación de objetos esto permite apilar múltiples decoradores sobre la carta base permitiendo componer comportamientos de manera flexible y en tiempo de ejecución.