EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Tijuana

Departamento de Sistemas y Computación

Nombre de la Materia:

Patrones de Diseño

Actividad:

Evaluación Unidad 2

Profesor:

Maribel Guerrero Luis

Alumno(s):

Jonathan Rafael Tamayo Ortiz

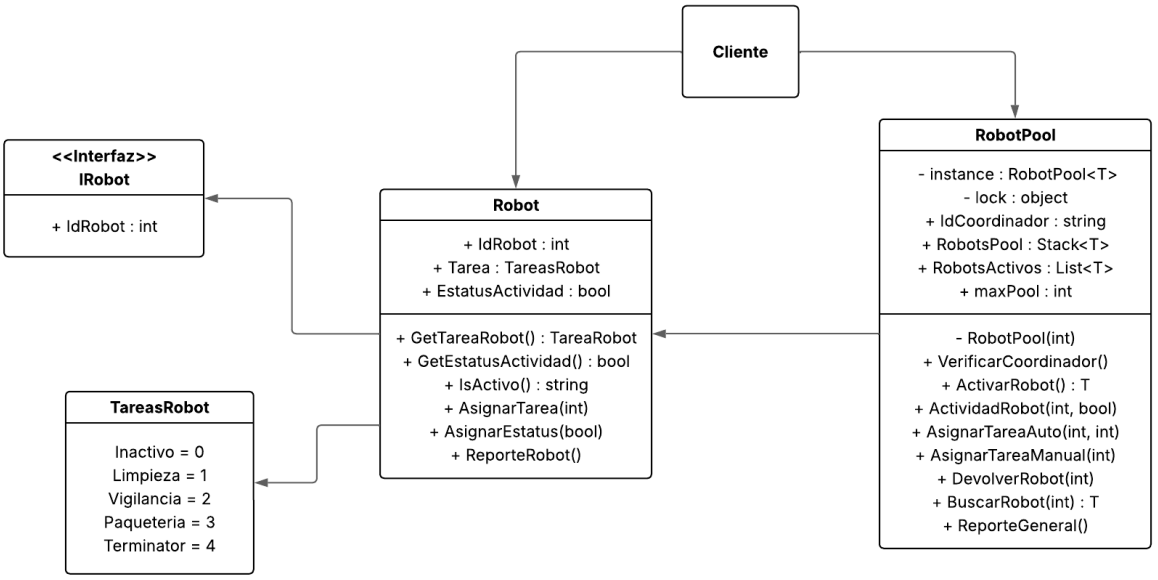
Fecha de entrega:

17/10/2025

Evaluación Unidad 2

La evaluación resuelve el problema de automatización y control de robots. Los patrones utilizados para esta tarea fueron Object Pool y Singleton.

Diagrama



Código

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EvaluacionU2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int opc = 0;
            int id = 0;

            do
              {   //Intenta crear una nuava instancia con cada ciclo. Singleton devuelve la ya existente
                RobotPool<Robot> coordinador = RobotPool<Robot>.Instance;
                Console.WriteLine("===Menu principal===\n");
                coordinador.VerificarCoordinador();
                coordinador.ReporteGeneral();
                Console.WriteLine("\n\nEliga una opcion:");
```

```
            Console.WriteLine("1.- Activar un robot");
            Console.WriteLine("2.- Desactivar un robot");
            Console.WriteLine("3.- Cambiar la tarea de un robot");
            Console.WriteLine("4.- Ver informe de un robot");
            Console.WriteLine("0.- Salir\n");
            int.TryParse(Console.ReadLine(), out opc);

            switch (opc)
            {
               case 1: //Activar robot
                  var robot1 = coordinador.ActivarRobot();
                   if (robot1 != null) { coordinador.AsignarTareaManual(robot1.IdRobot);
}
                  else { Console.WriteLine("No se pudo encontrar al robot"); }
                  break;
               case 2: //Devolver robot
                  Console.WriteLine("Ingrese el ID del robot que desea devolver:");
                  int.TryParse(Console.ReadLine(), out id);
                  coordinador.DevolverRobot(id);
                  break;
               case 3: //Cambiar la tarea de un robot
                     Console.WriteLine("Ingrese el ID del robot al que desea asignar una
nueva tarea:");
                  int.TryParse(Console.ReadLine(), out id);
                  var robot3 = coordinador.BuscarRobot(id);
                  if (!robot3.getEstatusActividad())
                  {
                      Console.WriteLine("El robot debe ser activado antes de asignarle
actividad");
                      break;
                   }
                   if (robot3 != null) { coordinador.AsignarTareaManual(robot3.IdRobot);
}
                  else { Console.WriteLine("No se pudo encontrar al robot"); }
                  break;
               case 4: //Muestra el estado de un solo robot
                  Console.WriteLine("Ingrese el ID del robot al que desea visualizar:");
                  int.TryParse(Console.ReadLine(), out id);
                  var robot4 = coordinador.BuscarRobot(id);
                  if (robot4 != null) { robot4.ReporteRobot(); }
                  else { Console.WriteLine("No se pudo encontrar al robot"); }
                  break;
               case 0: //Salir
                  break;
               default:
                  Console.WriteLine("Esa opcion no es valida.");
                  break;
            }
            Console.ReadKey();
            Console.Clear();
```

```
        } while (opc != 0);
      }
   }
}
—
public interface IRobot
{
   int IdRobot { get; set; } //Id de los robots
}
—
public class Robot : IRobot
{
   public int IdRobot { get; set; }
    public enum TareaRobot { Inactivo = 0, Limpieza = 1, Vigilancia = 2, Paqueteria =
3, Terminator = 4 }
   public TareaRobot Tarea;
     protected bool EstatusActividad = false; //El robot esta activo (true) o inactivo
(false)

    public TareaRobot GetTareaRobot() //Devuelve la tarea que esta realizando un
robot
   {
      return Tarea;
   }
   public bool getEstatusActividad() //Devuelve un valor booleano
   {
      return EstatusActividad;
   }
   public string isActivo() //Devuelve un valor string
   {
      if(getEstatusActividad() == true) //activo = (true) e inactivo = (false)
      {
         return "Activo";
      }
      return "Inactivo";
   }
   public void AsignarTarea(int tipo)
   {
       if (getEstatusActividad() == true) // Si el robot esta acivo, asigna la tarea que
corresponde
      {
         this.Tarea = (TareaRobot)tipo;
      }
      else // Si el robot esta inactivo, lo marca como 0
      {
         this.Tarea = (TareaRobot)0;
      }
   }
   public void AsignarEstatus(bool estatus)
   {
```

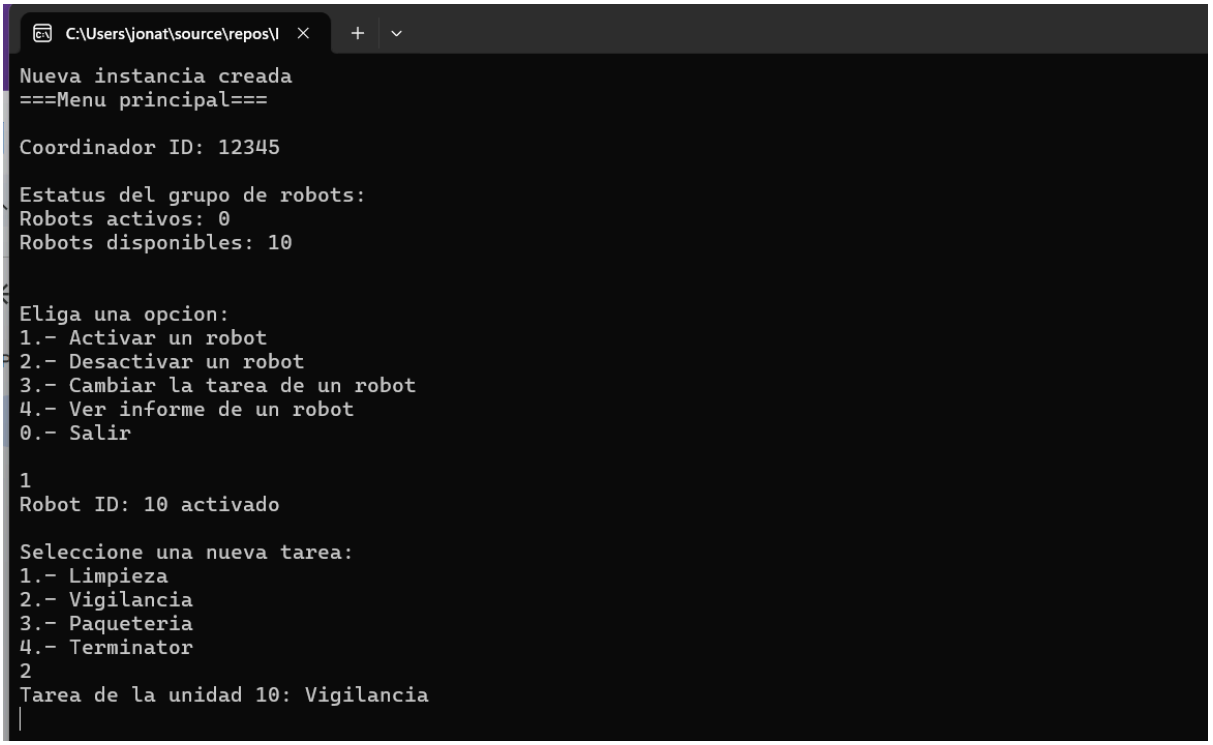
```
        EstatusActividad = estatus;
    }
    public void ReporteRobot()
    {
        Console.WriteLine($"\nRobot ID: {IdRobot}\nTarea: {GetTareaRobot()}\nEstatus:
{isActivo()}");
    }
}
—
public class RobotPool<T> where T : class, IRobot, new()
{   //Singleton
    private static RobotPool<T> _instance; //Una sola instancia de la piscina
    private static readonly object _lock = new object();
    public string IdCoordinador { get; set; }


    //Object pool
      public readonly Stack<T> RobotsPool = new Stack<T>(); //Stack de todos los
robots a la espera de instrucciones
    private readonly List<T> RobotsActivos = new List<T>(); //Lista de robots en uso
    private int maxPool { get; set; } //Tamaño del pool de robots

    private RobotPool(int maxPool)
    {
        IdCoordinador = "12345";
        this.maxPool = maxPool;
        //Instanciar los robots pertenecientes al pool
        for(int i = 1; i <= maxPool; i++)
        {
            var robot = new T();
            robot.IdRobot = i;
            RobotsPool.Push(robot);
            AsignarTareaAuto(robot.IdRobot, 0);
        }
    }

    public static RobotPool<T> Instance // Inicializa la instancia del coordinador
    {                                   // Lo cual inicializa la piscina de objetos
        get
        {
            if (_instance == null)
            {
                lock (_lock)
                {
                    if (_instance == null)
                    {
                        _instance = new RobotPool<T>(10);
                        Console.WriteLine("Nueva instancia creada");
                    }
                }
```

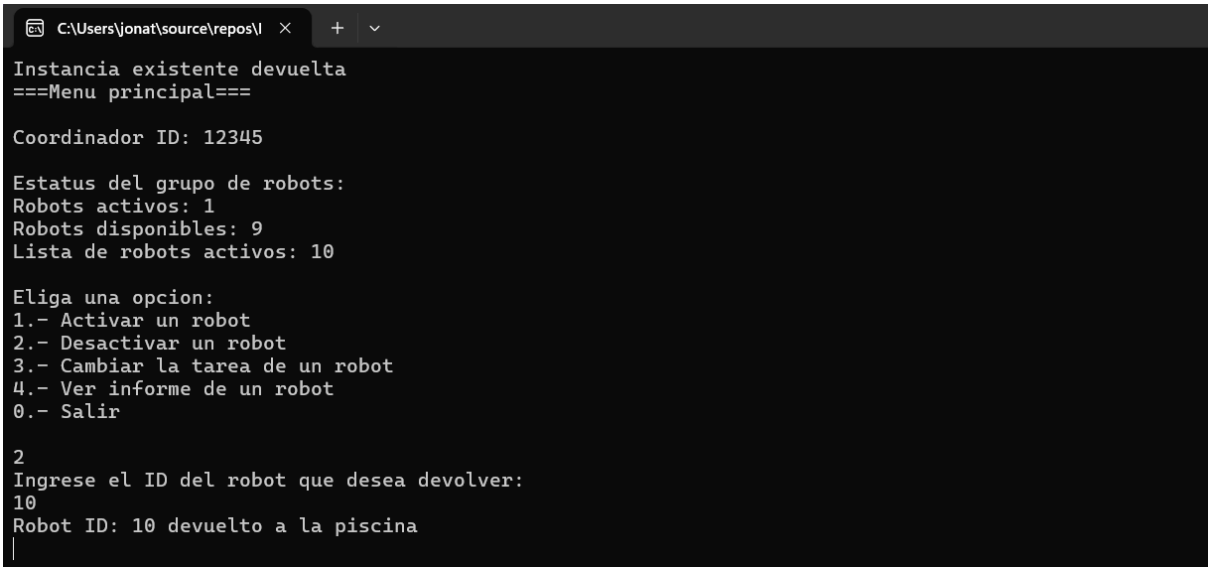
```
            }
            else
            {
                Console.WriteLine("Instancia existente devuelta");
            }
            return _instance;
        }
    }
    public void VerificarCoordinador()
    {
        Console.WriteLine($"Coordinador ID: {IdCoordinador}");
    }
    public T ActivarRobot() //Activa un robot
    {
        T robot;
        if(RobotsPool.Count > 0)
        {
            robot = RobotsPool.Pop();
            RobotsActivos.Add(robot);
            ActividadRobot(robot.IdRobot, true);
            Console.WriteLine($"Robot ID: {robot.IdRobot} activado");
            return robot;
        }
        Console.WriteLine("No es posible activar un robot en este momento");
        return null;
    }
    public void ActividadRobot(int id, bool estatus) //Marca un robot como activo
    {
        var robot = BuscarRobot(id);
        if(robot != null && robot is Robot r)
        {
            r.AsignarEstatus(estatus);
        }
    }
            public void AsignarTareaAuto(int id, int tarea) //La tarea se asigna
automaticamente por el sistema
    {
        var robot = BuscarRobot(id);
        if (robot != null && robot is Robot r)
        {
            r.AsignarTarea(tarea);
        }
    }
    public void AsignarTareaManual(int id) //La tarea es seleccionada por el usuario
    {
        Console.WriteLine("\nSeleccione una nueva tarea:\n" +
                          $"1.- Limpieza\n2.- Vigilancia\n3.- Paqueteria\n4.- Terminator");

          if (int.TryParse(Console.ReadLine(), out int tarea) && (tarea >= 0 && tarea <=
4))
```

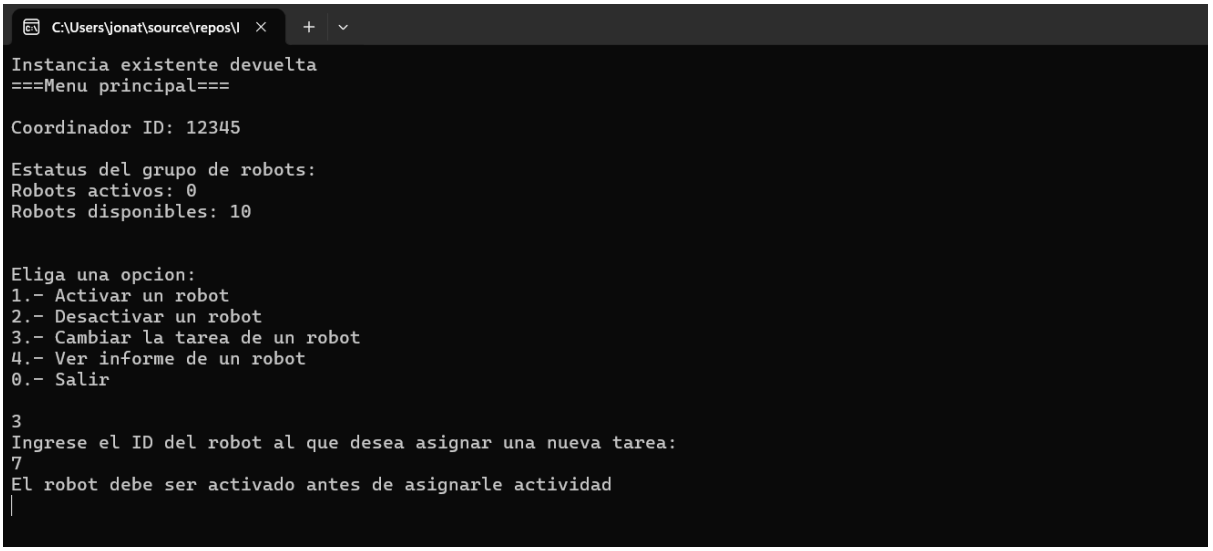
```
    {
        var robot = BuscarRobot(id);
        if (robot != null && robot is Robot r)
        {
            r.AsignarTarea(tarea);
            Console.WriteLine($"Tarea de la unidad {r.IdRobot}: {r.GetTareaRobot()}");
        }
        else
        {
            Console.WriteLine("No es posible asignar esa tarea.");
        }
    }
    else
    {
        Console.WriteLine("Opcion invalida.");
    }
}
public void DevolverRobot(int Id) //Devuelve el robot al pool
{
    var robot = RobotsActivos.FirstOrDefault(r => r.IdRobot == Id);
    if (robot != null)
    {
        RobotsActivos.Remove(robot);
        RobotsPool.Push(robot);
        ActividadRobot(robot.IdRobot, false);
        AsignarTareaAuto(robot.IdRobot, 0);
        Console.WriteLine($"Robot ID: {robot.IdRobot} devuelto a la piscina");
    }
    else
    {
        Console.WriteLine("Robot no encontrado");
    }
}
public T BuscarRobot(int Id)
{
    return RobotsPool.Concat(RobotsActivos).FirstOrDefault(r => r.IdRobot == Id);
}
public void ReporteGeneral()
{
    Console.WriteLine("\nEstatus del grupo de robots:\n" +
        $"Robots activos: {RobotsActivos.Count}\n" +
        $"Robots disponibles: {RobotsPool.Count}");
    if (RobotsActivos.Count > 0)
    {   // Muestra los robots que estan realizando alguna tarea, si los hay
        Console.Write("Lista de robots activos: ");
        foreach(var r in RobotsActivos.Cast<Robot>())
        {
            Console.Write(r.IdRobot + "\t");
        }
    }
```

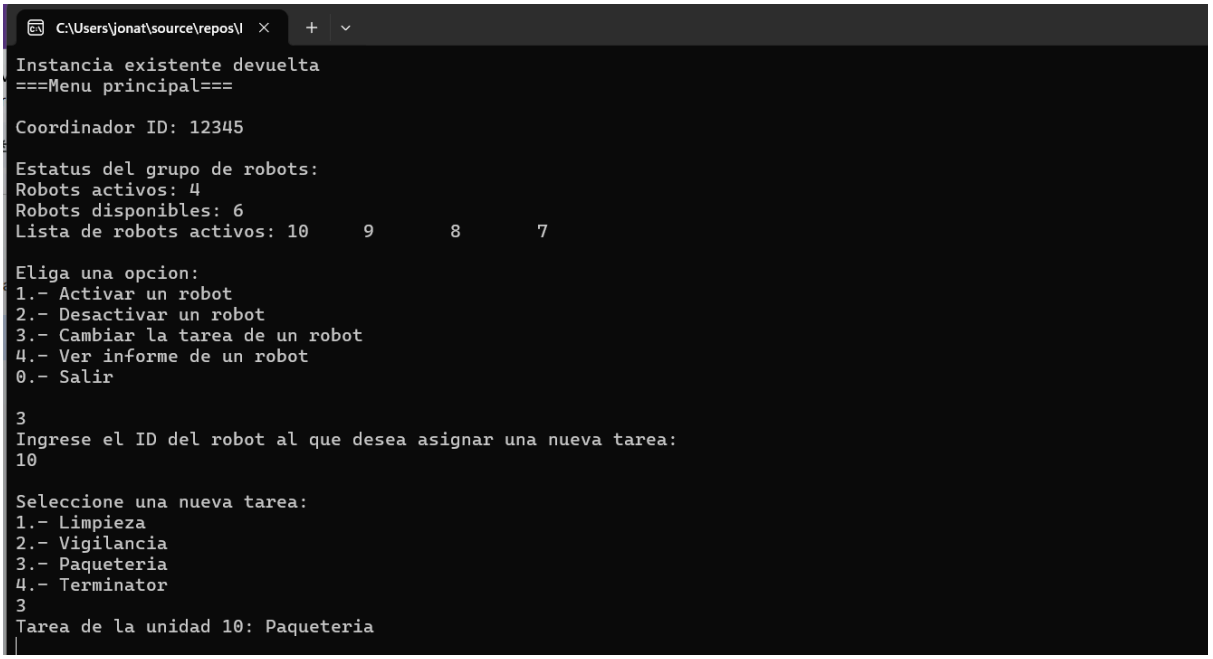
```
    }
}
```

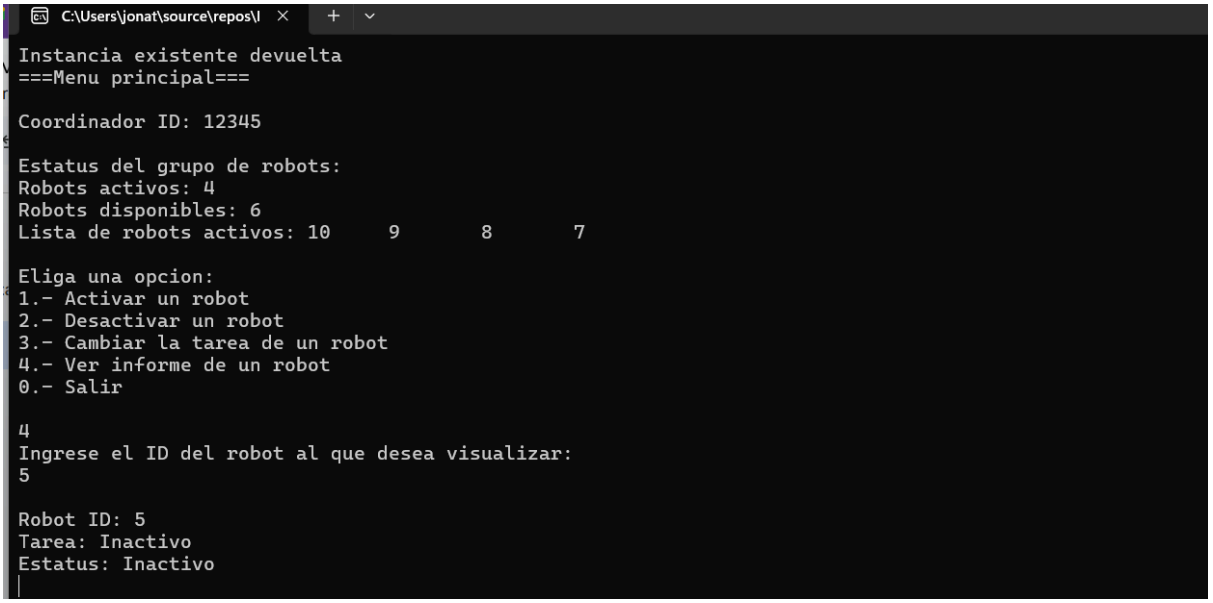
Capturas de pantalla

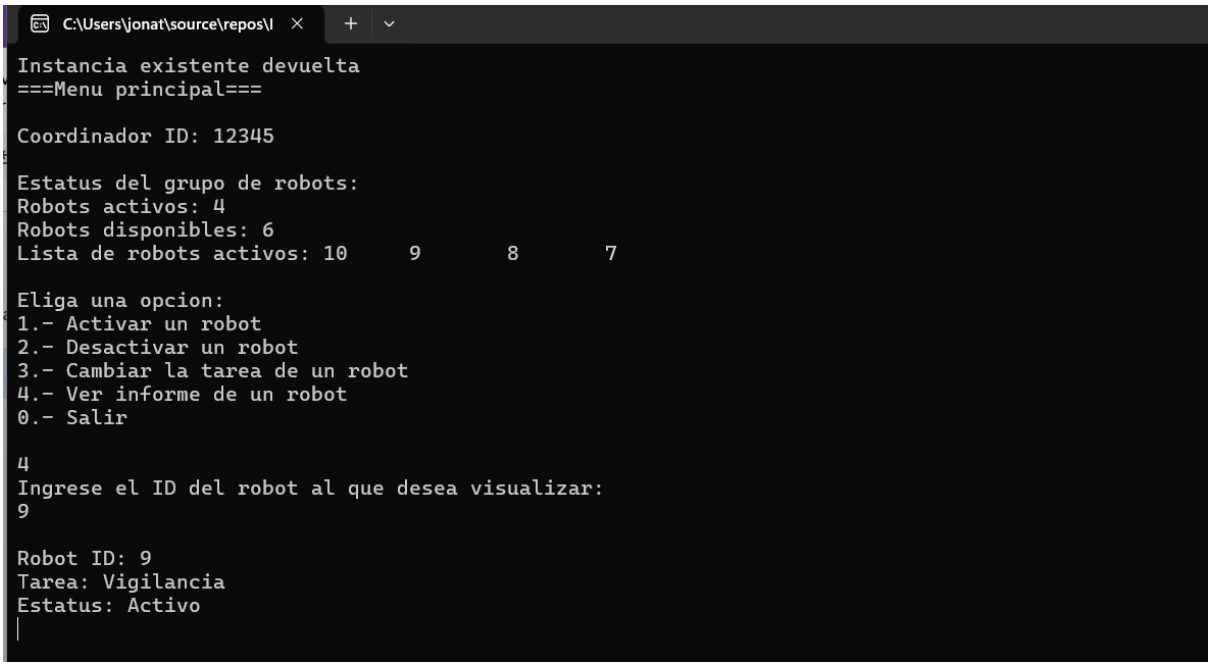



C:\Users\jonat\source\repos\I
Instancia existente devuelta
===Menu principal===
Coordinador ID: 12345
Estatus del grupo de robots:
Robots activos: 1
Robots disponibles: 9
Lista de robots activos: 10
Eliga una opcion:
1.- Activar un robot
2.- Desactivar un robot
3.- Cambiar la tarea de un robot
4.- Ver informe de un robot
0.- Salir
2
Ingrese el ID del robot que desea devolver:
10
Robot ID: 10 devuelto a la piscina


C:\Users\jonat\source\repos\I
Instancia existente devuelta
===Menu principal===
Coordinador ID: 12345
Estatus del grupo de robots:
Robots activos: 0
Robots disponibles: 10
Eliga una opcion:
1.- Activar un robot
2.- Desactivar un robot
3.- Cambiar la tarea de un robot
4.- Ver informe de un robot
0.- Salir
3
Ingrese el ID del robot al que desea asignar una nueva tarea:
7
El robot debe ser activado antes de asignarle actividad


C:\Users\jonat\source\repos\I
Instancia existente devuelta
===Menu principal===
Coordinador ID: 12345
Estatus del grupo de robots:
Robots activos: 4
Robots disponibles: 6
Lista de robots activos: 10 9 8 7
Eliga una opcion:
1.- Activar un robot
2.- Desactivar un robot
3.- Cambiar la tarea de un robot
4.- Ver informe de un robot
0.- Salir
3
Ingrese el ID del robot al que desea asignar una nueva tarea:
10
Seleccione una nueva tarea:
1.- Limpieza
2.- Vigilancia
3.- Paqueteria
4.- Terminator
3
Tarea de la unidad 10: Paqueteria





Conclusión

El uso de patrones Object Pool y Singleton nos permite asegurarnos de que solo existe una sola instancia del coordinador que le pueda dar instrucciones a los robots del object pool. Al combinar estos patrones, también nos aseguramos de que estamos utilizando una sola piscina de objetos. Adicionalmente, el uso de pool nos permite tener un número determinado de robots a los que darle instrucciones, y nos podemos ahorrar el costo de crear y destruir nuevos objetos.