İSİM : NATHANAELLE BOPTI NGAH BONG

OGRENCİ NUMERASI : 24080410150

İkili Arama Ağacı (BST) ile Forma Numarası Kayıt Sistemi

```
using System;
using System.Collections.Generic;
using System.Text;

namespace SportsClubBST
{
   public class OyuncuNode
    {
        public int FormaNumerasi;
        public string Ad;
        public string Soyad;
        public OyuncuNode Left;
        public OyuncuNode Right;

        public OyuncuNode(int formanumerasi, string ad, string soyad)
        {
            FormaNumerasi = formanumerasi;
             Ad = soyad;
            Soyad = soyad;
            Left = null;
            Right = null;
        }
    }
    class PlayerBST
    {
        private OyuncuNode root;

        public PlayerBST()
        {
            root = null;
        }

     //Ekleme
        public void Insert(int jerseyNumber, string firstName, string lastName)
        {
            OyuncuNode newNode = new OyuncuNode(jerseyNumber, firstName, lastName);

            if (root == null)
            {
                root = newNode;
                Console.WriteLine($"Goalkeeper {firstName} {lastName}
(#{jerseyNumber}) kok olarak eklendi.");
            }
            else
```

```
            {
                InsertRecursive(root, newNode);
                Console.WriteLine($"Player {firstName} {lastName} (#{jerseyNumber})
eklendi.");
            }
        }

        private void InsertRecursive(OyuncuNode current, OyuncuNode newNode)
        {
            if (newNode.FormaNumerasi < current.FormaNumerasi)
            {
                if (current.Left == null)
                    current.Left = newNode;
                else
                    InsertRecursive(current.Left, newNode);
            }
            else if (newNode.FormaNumerasi > current.FormaNumerasi)
            {
                if (current.Right == null)
                    current.Right = newNode;
                else
                    InsertRecursive(current.Right, newNode);
            }
            else
            {
                Console.WriteLine($"Jersey number {newNode.FormaNumerasi} already
exists!");
            }
        }


        public OyuncuNode Search(int jerseyNumber)
        {
            return SearchRecursive(root, jerseyNumber);
        }

        private OyuncuNode SearchRecursive(OyuncuNode current, int jerseyNumber)
        {
            if (current == null || current.FormaNumerasi == jerseyNumber)
                return current;

            if (jerseyNumber < current.FormaNumerasi)
                return SearchRecursive(current.Left, jerseyNumber);

            return SearchRecursive(current.Right, jerseyNumber);
        }

        public void Delete(int jerseyNumber)
        {
            root = DeleteRecursive(root, jerseyNumber);
        }

        private OyuncuNode DeleteRecursive(OyuncuNode current, int jerseyNumber)
        {
            if (current == null)
                return current;

            if (jerseyNumber < current.FormaNumerasi)
```

```
            current.Left = DeleteRecursive(current.Left, jerseyNumber);
        else if (jerseyNumber > current.FormaNumerasi)
            current.Right = DeleteRecursive(current.Right, jerseyNumber);
        else
        {
            // Node with one child or no child
            if (current.Left == null)
                return current.Right;
            else if (current.Right == null)
                return current.Left;

            // Node with two children: Get inorder successor
            OyuncuNode temp = FindMinNode(current.Right);
            current.FormaNumerasi = temp.FormaNumerasi;
            current.Ad = temp.Ad;
            current.Soyad = temp.Soyad;
            current.Right = DeleteRecursive(current.Right, temp.FormaNumerasi);
        }
        return current;
    }

    // 4. Traversals
    public void PreorderTraversal()
    {
        Console.Write("Preorder: ");
        PreorderRecursive(root);
        Console.WriteLine();
    }

    private void PreorderRecursive(OyuncuNode node)
    {
        if (node != null)
        {
            Console.Write($"#{node.FormaNumerasi} ({node.Ad}) → ");
            PreorderRecursive(node.Left);
            PreorderRecursive(node.Right);
        }
    }

    public void InorderTraversal()
    {
        Console.Write("Inorder: ");
        InorderRecursive(root);
        Console.WriteLine();
    }

    private void InorderRecursive(OyuncuNode node)
    {
        if (node != null)
        {
            InorderRecursive(node.Left);
            Console.Write($"#{node.FormaNumerasi} ({node.Ad}) → ");
            InorderRecursive(node.Right);
        }
    }

    public void PostorderTraversal()
    {
```

```
    Console.Write("Postorder: ");
    PostorderRecursive(root);
    Console.WriteLine();
}

private void PostorderRecursive(OyuncuNode node)
{
    if (node != null)
    {
        PostorderRecursive(node.Left);
        PostorderRecursive(node.Right);
        Console.Write($"#{node.FormaNumerasi} ({node.Ad}) → ");
    }
}

public void LevelOrderTraversal()
{
    Console.Write("Level Order: ");
    if (root == null) return;

    Queue<OyuncuNode> queue = new Queue<OyuncuNode>();
    queue.Enqueue(root);

    while (queue.Count > 0)
    {
        OyuncuNode current = queue.Dequeue();
        Console.Write($"#{current.FormaNumerasi} ({current.Ad}) → ");

        if (current.Left != null)
            queue.Enqueue(current.Left);
        if (current.Right != null)
            queue.Enqueue(current.Right);
    }
    Console.WriteLine();
}

// 5. Find maximum jersey number
public OyuncuNode FindMax()
{
    if (root == null) return null;
    return FindMaxRecursive(root);
}

private OyuncuNode FindMaxRecursive(OyuncuNode node)
{
    while (node.Right != null)
        node = node.Right;
    return node;
}

// 6. Find minimum jersey number
public OyuncuNode FindMin()
{
    if (root == null) return null;
    return FindMinNode(root);
}

private OyuncuNode FindMinNode(OyuncuNode node)
```

```
        {
            while (node.Left != null)
                node = node.Left;
            return node;
        }

        // Helper method to display player info
        public void DisplayPlayerInfo(OyuncuNode player)
        {
            if (player != null)
                Console.WriteLine($"Found: #{player.FormaNumerasi} - {player.Ad}
{player.Soyad}");
            else
                Console.WriteLine("Player not found!");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            PlayerBST team = new PlayerBST();

            Console.WriteLine("=== SPORTS CLUB PLAYER MANAGEMENT SYSTEM ===\n");

            // Add goalkeeper first (becomes root)
            team.Insert(1, "David", "Goalie");  // Goalkeeper

            // Add other players
            team.Insert(10, "Lionel", "Playmaker");
            team.Insert(7, "Cristiano", "Striker");
            team.Insert(4, "Sergio", "Defender");
            team.Insert(9, "Robert", "Forward");
            team.Insert(5, "Virgil", "Defender");

            Console.WriteLine("\n--- Player Traversals ---");
            team.PreorderTraversal();
            team.InorderTraversal();
            team.PostorderTraversal();
            team.LevelOrderTraversal();

            Console.WriteLine("\n--- Search Operations ---");
            team.DisplayPlayerInfo(team.Search(7));
            team.DisplayPlayerInfo(team.Search(99));  // Non-existent

            Console.WriteLine("\n--- Min/Max Jersey Numbers ---");
            var minPlayer = team.FindMin();
            var maxPlayer = team.FindMax();

            if (minPlayer != null)
                Console.WriteLine($"Minimum Jersey: #{minPlayer.FormaNumerasi} -
{minPlayer.Ad}");
            if (maxPlayer != null)
                Console.WriteLine($"Maximum Jersey: #{maxPlayer.FormaNumerasi} -
{maxPlayer.Ad}");
```

```
            Console.WriteLine("\n--- Delete Player #4 ---");
            team.Delete(4);
            Console.WriteLine("After deletion:");
            team.InorderTraversal();
        }
    }
}
```

```
Goalkeeper David Goalie (#1) kok olarak eklendi.
Player Lionel Playmaker (#10) eklendi.
Player Cristiano Striker (#7) eklendi.
Player Sergio Defender (#4) eklendi.
Player Robert Forward (#9) eklendi.
Player Virgil Defender (#5) eklendi.

--- Player Traversals ---
Preorder: #1 (Goalie) → #10 (Playmaker) → #7 (Striker) → #4 (Defender) → #5 (Defender) → #9 (Forward) →
Inorder: #1 (Goalie) → #4 (Defender) → #5 (Defender) → #7 (Striker) → #9 (Forward) → #10 (Playmaker) →
Postorder: #5 (Defender) → #4 (Defender) → #9 (Forward) → #7 (Striker) → #10 (Playmaker) → #1 (Goalie) →
Level Order: #1 (Goalie) → #10 (Playmaker) → #7 (Striker) → #4 (Defender) → #9 (Forward) → #5 (Defender) →

--- Search Operations ---
Found: #7 - Striker Striker
Player not found!

--- Min/Max Jersey Numbers ---
Minimum Jersey: #1 - Goalie
Maximum Jersey: #10 - Playmaker

--- Delete Player #4 ---
After deletion:
Inorder: #1 (Goalie) → #5 (Defender) → #7 (Striker) → #9 (Forward) → #10 (Playmaker) →
```