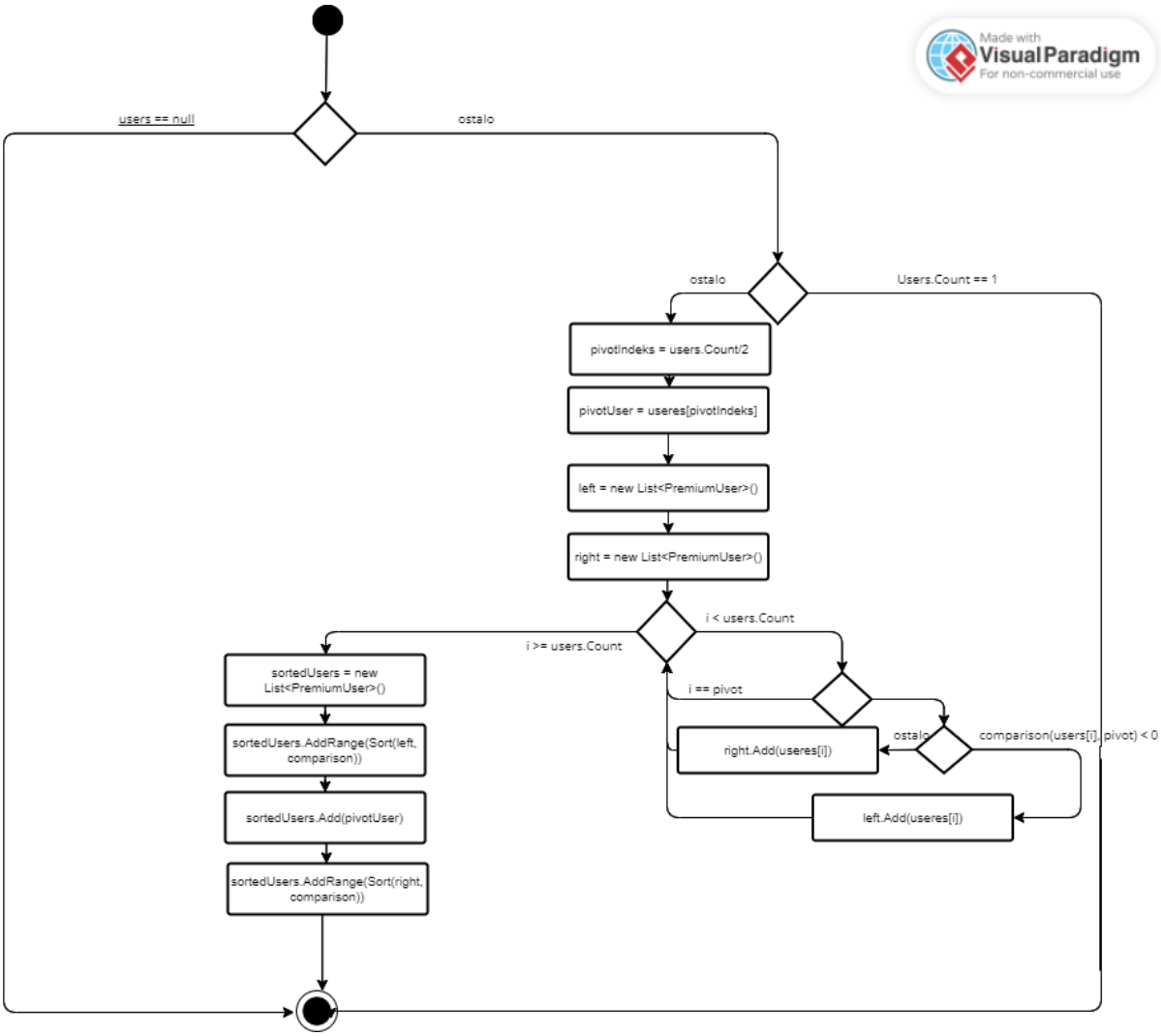
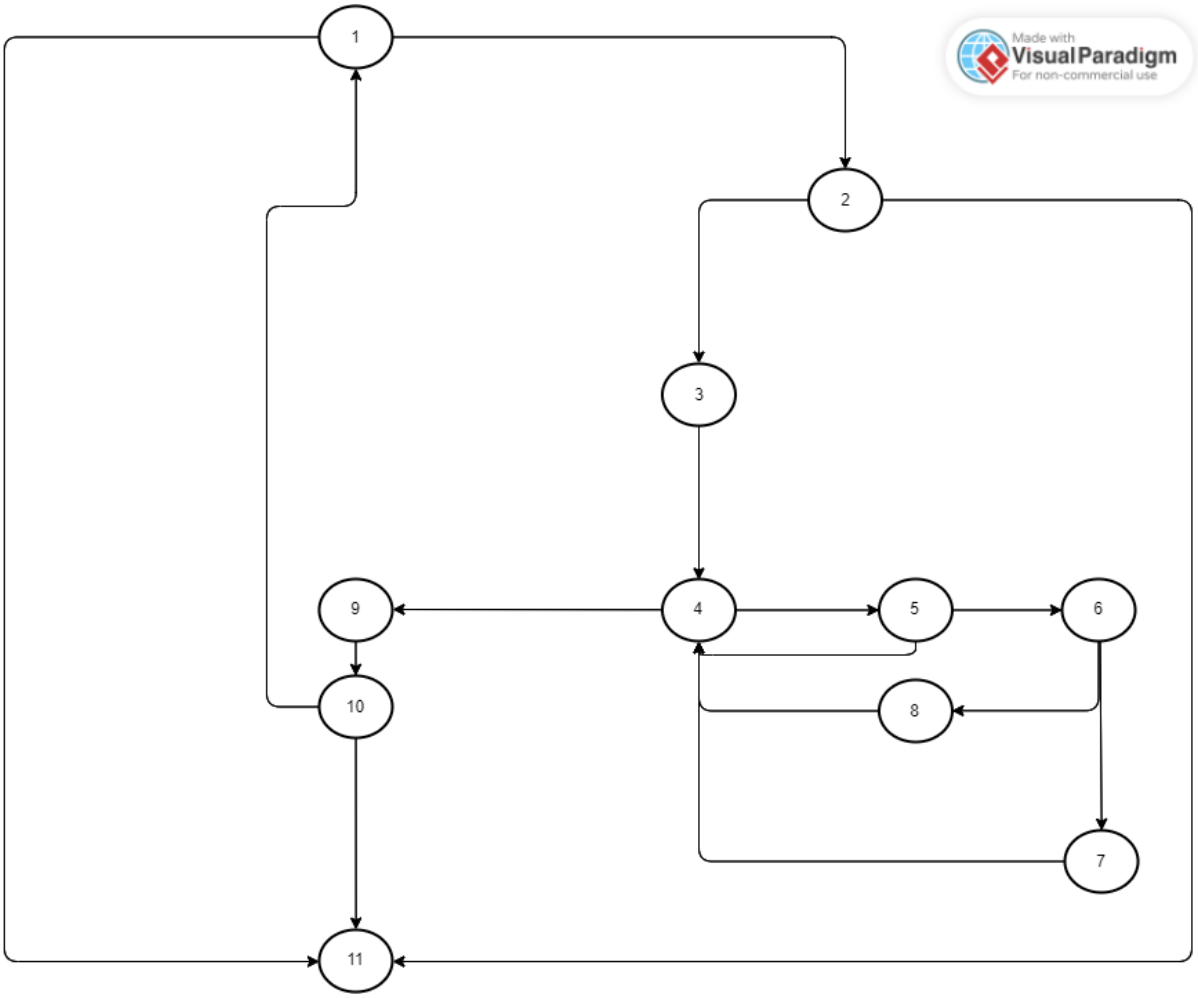
Metoda koja je testirana:

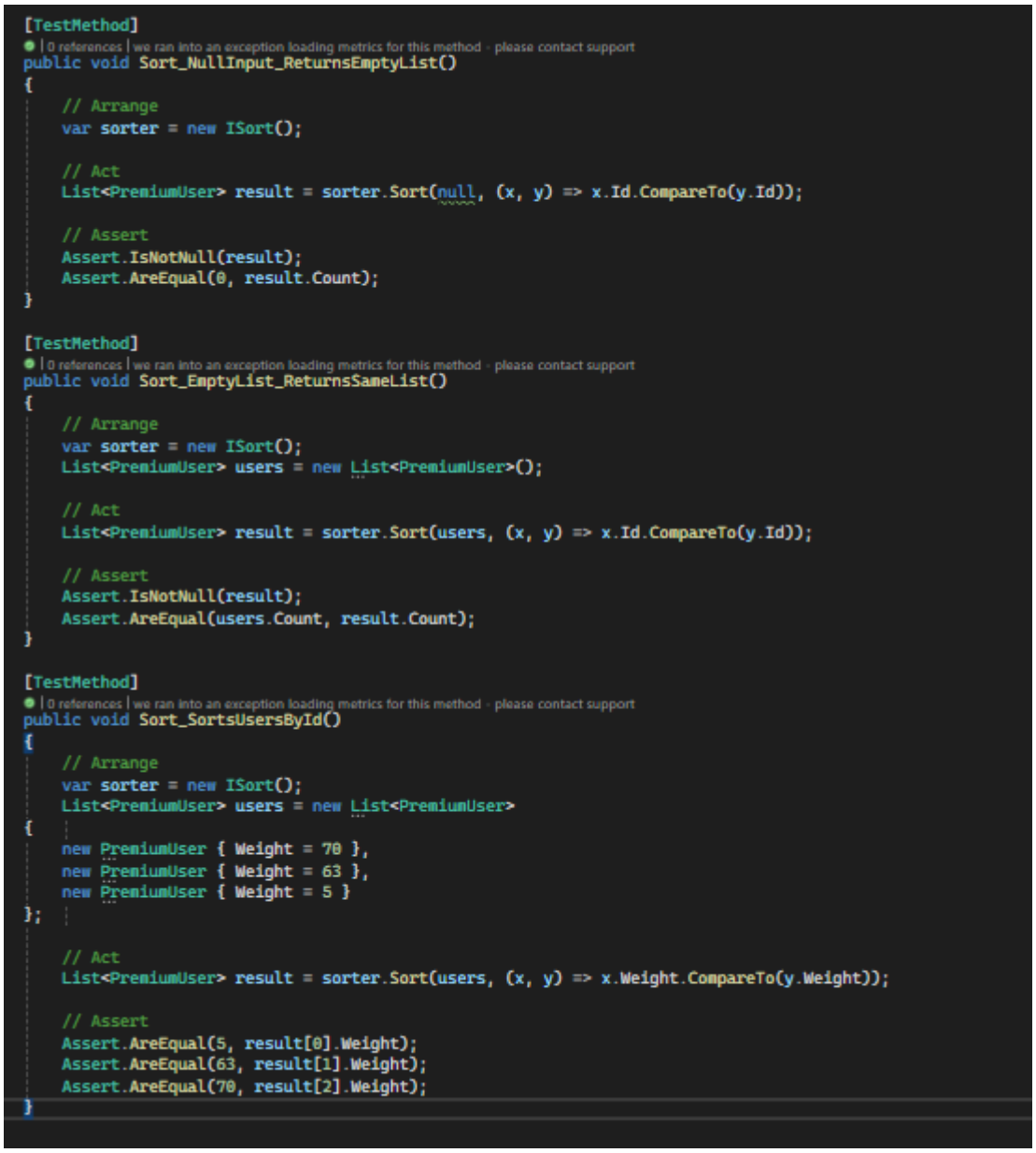
```
/// returns a sorted list of PremiumUser objects.
8 references | we ran into an exception loading metrics for this method - please contact support | 3/3 passing
public List<PremiumUser> Sort(List<PremiumUser> users, Func<PremiumUser, PremiumUser, int> comparison)
{
        if (users == null)
        {
            return new List<PremiumUser>();
        }

        if (users.Count <= 1)
        {
            return users;
        }

        int pivotIndex = users.Count / 2;
        PremiumUser pivotUser = users[pivotIndex];

        List<PremiumUser> left = new List<PremiumUser>();
        List<PremiumUser> right = new List<PremiumUser>();

        for (int i = 0; i < users.Count; i++)
        {
            if (i == pivotIndex)
            {
                continue;
            }

            int comparisonResult = comparison(users[i], pivotUser);

            if (comparisonResult < 0)
            {
                left.Add(users[i]);
            }
            else
            {
                right.Add(users[i]);
            }
        }

        List<PremiumUser> sortedUsers = new List<PremiumUser>();
        sortedUsers.AddRange(Sort(left, comparison));
        sortedUsers.Add(pivotUser);
        sortedUsers.AddRange(Sort(right, comparison));

        return sortedUsers;
}
```


users == null
ostalo
ostalo
Users.Count == 1
pivotIndeks = users.Count/2
pivotUser = useres[pivotIndeks]
left = new List<PremiumUser>()
right = new List<PremiumUser>()
i < users.Count
i >= users.Count
sortedUsers = new List<PremiumUser>()
sortedUsers.AddRange(Sort(left, comparison))
sortedUsers.Add(pivotUser)
sortedUsers.AddRange(Sort(right, comparison))
i == pivot
right.Add(useres[i])
ostalo
comparison(users[i], pivot) < 0
left.Add(useres[i])



1	1, 2, 3, 4, 5, 6, 7, 4, 5, 6, 8, 4, 9, 10, 11
2	1, 2, 3, 4, 5, 6, 7, 4, 9, 10, 11
3	1, 2, 3, 4, 5, 6, 8, 4, 9, 10, 11
4	1, 2, 3, 4, 5, 4, 9, 10, 11
5	1, 2, 3, 4, 9, 10, 11
6	1, 2, 11
7	1, 11

Testovi:



```
[TestMethod]
0 references | we ran into an exception loading metrics for this method - please contact support
public void Sort_NullInput_ReturnsEmptyList()
{
    // Arrange
    var sorter = new ISort();

    // Act
    List<PremiumUser> result = sorter.Sort(null, (x, y) => x.Id.CompareTo(y.Id));

    // Assert
    Assert.IsNotNull(result);
    Assert.AreEqual(0, result.Count);
}

[TestMethod]
0 references | we ran into an exception loading metrics for this method - please contact support
public void Sort_EmptyList_ReturnsSameList()
{
    // Arrange
    var sorter = new ISort();
    List<PremiumUser> users = new List<PremiumUser>();

    // Act
    List<PremiumUser> result = sorter.Sort(users, (x, y) => x.Id.CompareTo(y.Id));

    // Assert
    Assert.IsNotNull(result);
    Assert.AreEqual(users.Count, result.Count);
}

[TestMethod]
0 references | we ran into an exception loading metrics for this method - please contact support
public void Sort_SortsUsersById()
{
    // Arrange
    var sorter = new ISort();
    List<PremiumUser> users = new List<PremiumUser>
{
    new PremiumUser { Weight = 70 },
    new PremiumUser { Weight = 63 },
    new PremiumUser { Weight = 5 }
};

    // Act
    List<PremiumUser> result = sorter.Sort(users, (x, y) => x.Weight.CompareTo(y.Weight));

    // Assert
    Assert.AreEqual(5, result[0].Weight);
    Assert.AreEqual(63, result[1].Weight);
    Assert.AreEqual(70, result[2].Weight);
}
```

Proces testiranja je pažljivo osmišljen kako bi uključio širok spektar strategija. Testovi su pažljivo planirani i implementirani kako bi obuhvatili sve nivoe koda - od osnovnih iskaza i grana, pa sve do petlji, toka podataka i čak svih mogućih puteva unutar programa. Ovaj sveobuhvatan pristup testiranju osigurava da softver prolazi kroz rigorozne scenarije i omogućava detekciju potencijalnih grešaka na svakom nivou implementacije.