White box testiranje – izvještaj

Controllers > StudentController > `public List<Student> BubbleSort(List<Student> Index)`

```
public List<Student> BubbleSort(List<Student> Index)
{
    //Bubble sort of index by name
    for (int i = 0; i < Index.Count - 1; i++)
    {
        for (int j = 0; j < Index.Count - i - 1; j++)
        {
            if (Index[j].FirstName.CompareTo(Index[j + 1].FirstName) > 0)
            {
                (Index[j + 1], Index[j]) = (Index[j], Index[j + 1]);
            }
        }
    }
    return Index;
}
```

1. Line coverage:

```
[TestMethod]
public void BubbleSort_EveryLineCovered()
{
    var students = new List<Student>
    {
        new Student { Index = 2, FirstName = "John" },
        new Student { Index = 1, FirstName = "Ana" }
    };
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(1, students[0].Index);
    Assert.AreEqual(2, students[1].Index);
}
```

2. Branch coverage:

```
[TestMethod]
public void BubbleSort_EveryBranchCovered()
{
    var students = new List<Student>
    {
        new Student { Index = 1, FirstName = "John" },
        new Student { Index = 2, FirstName = "Bob"},
        new Student { Index = 4, FirstName = "Alice"},
        new Student { Index = 3, FirstName = "Meho"}
    };
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(1, students[0].Index);
    Assert.AreEqual(2, students[1].Index);
    Assert.AreEqual(3, students[2].Index);
    Assert.AreEqual(4, students[3].Index);
}
```

3. Condition coverage:

```
[TestMethod]
public void BubbleSort_AllConditionsCovered()
{
    var students = new List<Student>
    {
        new Student { Index = 2 },
        new Student { Index = 187 },
        new Student { Index = 1 },
        new Student { Index = 100 },
        new Student { Index = 13 },
        new Student { Index = 47 }
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

4. modifikovani uslov / odluka obuhvat

```
[TestMethod]
public void BubbleSort_WhenIfIsTrue()
{
    List<Student> students = new List<Student>
    {
        new Student { Index = 3, FirstName = "John" },
        new Student { Index = 2, FirstName = "Alice"},
        new Student { Index = 1, FirstName = "Bob"}
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

```
[TestMethod]
public void BubbleSort_WhenIfIsFalse()
{
    List<Student> students = new List<Student>
    {
        new Student { Index = 1, FirstName = "John" },
        new Student { Index = 2, FirstName = "Alice"},
        new Student { Index = 3, FirstName = "Bob"}
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

5. loop coverage

```
[TestMethod]
public void BubbleSort_testCase1()
{
    List<Student> students = new List<Student>
    {
        // minimalan broj prolaza vanjsku i preskoči unutrašnjost unutrašnje
        new Student { Index = 2, FirstName = "John" }
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

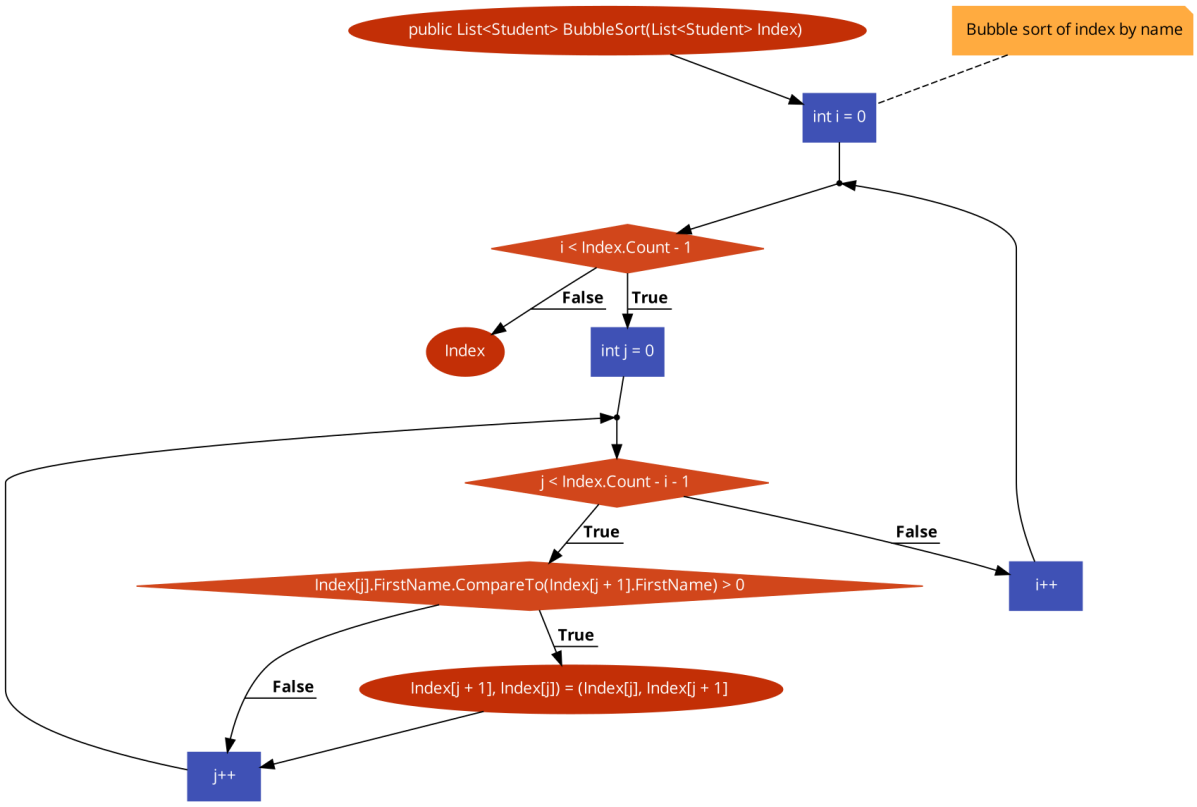
```
[TestMethod]
public void BubbleSort_testCase2()
{
    List<Student> students = new List<Student>
    {
        // minimalan broj prolaza vanjsku i 1 prolaz kroz unutrašnju
        new Student { Index = 2, FirstName = "John" },
        new Student { Index = 1, FirstName = "Bob" }
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

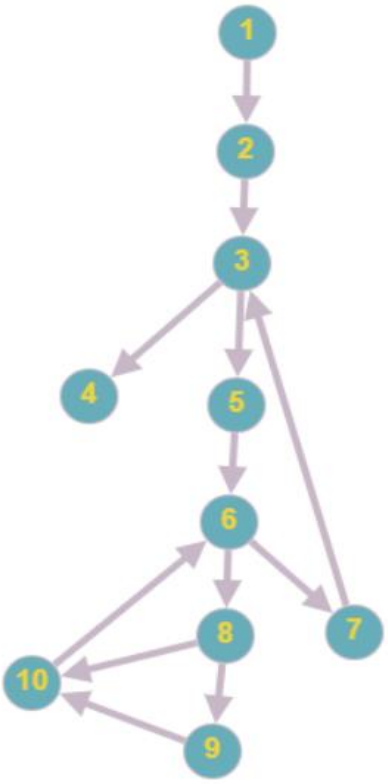
```
[TestMethod]
public void BubbleSort_testCase3()
{
    List<Student> students = new List<Student>
    {
        //minimalan broj prolaza vanjsku i 2 prolaza kroz unutrašnju
        new Student { Index = 3, FirstName = "John" },
        new Student { Index = 2, FirstName = "Bob"},
        new Student { Index = 1, FirstName = "Alice"}
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

```
[TestMethod]
public void BubbleSort_testCase4()
{
    List<Student> students = new List<Student>
    {
        //random broj prolazaka kroz petlju (nesortirana lista)
        new Student { Index = 2, FirstName = "Zineta" },
        new Student { Index = 3, FirstName = "Šarafeta" },
        new Student { Index = 6, FirstName = "Šeherzada" },
        new Student { Index = 1, FirstName = "Hamo"},
        new Student { Index = 4, FirstName = "Đulzulejha"},
        new Student { Index = 5, FirstName = "Hanifa"}
    };
    students = StudentController.BubbleSort(students);
    for (int i = 1; i < students.Count; i++)
    {
        Assert.IsTrue(students[i - 1].Index <= students[i].Index);
    }
}
```

```
[TestMethod]
public void BubbleSort_testCase5()
{
    //preskoči izvršavanje vanjske petlje
    List<Student> students = new List<Student>();
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(0, students.Count);
}
```

6. Path coverage





Broj puta	Put
1.	1-2-3-4
2.	1-2-3-5-6-8-9-10-6-7-3-4
3.	1-2-3-5-6-8-10-6-7-3-4
4.	1-2-3-5-6-7-3-4

Četiri različita puta:

```
[TestMethod]
public void BubbleSort_Path1()
{
    List<Student> students = new List<Student>();
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(0, students.Count);
}
```

```
[TestMethod]
public void BubbleSort_Path2()
{
    var students = new List<Student>
    {
        new Student { Index = 2, FirstName = "John" },
        new Student { Index = 1, FirstName = "Ana" }
    };
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(1, students[0].Index);
    Assert.AreEqual(2, students[1].Index);
}
```

```
[TestMethod]
public void BubbleSort_Path3()
{
    var students = new List<Student>
    {
        new Student { Index = 1, FirstName = "Anna" },
        new Student { Index = 2, FirstName = "John" }
    };
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(1, students[0].Index);
    Assert.AreEqual(2, students[1].Index);
}
```

```
[TestMethod]
public void BubbleSort_Path4()
{
    var students = new List<Student>
    {
        new Student { Index = 1, FirstName = "Anna" }
    };
    students = StudentController.BubbleSort(students);
    Assert.AreEqual(1, students[0].Index);
}
```