Selma Ličina 19148

White box testiranje – izvještaj

Metoda GetExamResults(string link) u SheetsFacade:

```
public static Dictionary<int, double> GetExamResults(string link)
{
    string id = ExtractSpreadsheetId(link);

    _manager = new GoogleSheetsManager(id);

    var stringResults = _manager.GetExam();

    var results = new Dictionary<int, double>();


    foreach (var kvp in stringResults)
    {
        if (int.TryParse(kvp.Key, out int parsedKey) && double.TryParse(kvp.Value,
out double parsedValue))
        {
            results.Add(parsedKey, parsedValue);
        }
        else
        {
            throw new Exception("Incorrectly formated data in cells!");
        }
    }

    return results;
}
```

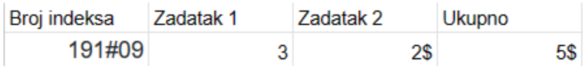
*Line Coverage

TC1:

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void GetExamResults_InvalidLink_ReturnsEmptyResults()
{
    // Arrange
    string link = "invalid link";

    // Act
    var results = SheetsFacade.GetExamResults(link);
}
```

Selma Ličina 19148

```
TC2:

[TestMethod]
public void GetExamResults_ValidLink_ReturnsResults()
{
    // Arrange
    string link =
"https://docs.google.com/spreadsheets/d/1vErXCyteFukR0smnSF8jN7nVhZznzaz2tNJo8pouWjU
/edit#gid=0";

    // Act
    var results = SheetsFacade.GetExamResults(link);

    // Assert
    Assert.IsNotNull(results);
    Assert.AreEqual(2, results.Count);
    Assert.IsTrue(results.ContainsKey(19109));
    Assert.IsTrue(results.ContainsKey(19163));
    Assert.AreEqual(10, results[19163]);
    Assert.AreEqual(5, results[19109]);
}


TC3:

[TestMethod]
[ExpectedException(typeof(Exception))]
public void GetExamResults_WrongFormat_ThrowsException()
{
    // Arrange
    string link =
"https://docs.google.com/spreadsheets/d/1xEbVjwSOi13Z_JS38Q9jjgRI2ePXr3xI4Ua10pixBS8
/edit?usp=sharing";

    // Act
    var results = SheetsFacade.GetExamResults(link);
}




*Branch Coverage

TC1:
```

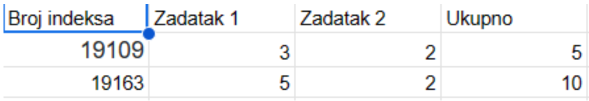


Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
191#09	3	2$	5$

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void GetExamResults_BothTryParseFail_ThrowsException()
{
    // Arrange
    string link = "
https://docs.google.com/spreadsheets/d/1cxJwq1Zp0MhPKsNVBEBgORfIINoZQaiJf8PdTt7bBgQ/
edit?usp=sharing";
```

```
    // Act
    var results = SheetsFacade.GetExamResults(link);
}
```

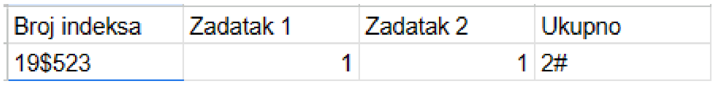
TC2:



Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19109	3	2	5
19163	5	2	10

```
[TestMethod]
public void GetExamResults_ValidNumbers_ReturnsResults()
{
    // Arrange
    string link =
"https://docs.google.com/spreadsheets/d/1vErXCyteFukR0smnSF8jN7nVhZznzaz2tNJo8pouWjU/edit#gid=0";

    // Act
    var results = SheetsFacade.GetExamResults(link);

    // Assert
    Assert.IsNotNull(results);
    Assert.AreEqual(2, results.Count);
    Assert.IsTrue(results.ContainsKey(19109));
    Assert.IsTrue(results.ContainsKey(19163));
    Assert.AreEqual(10, results[19163]);
    Assert.AreEqual(5, results[19109]);
}
```

TC3:

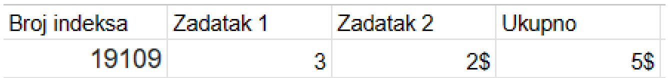


Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19$523	1	1	2#

```
[TestMethod]
[ExpectedException(typeof(Exception))]
public void GetExamResults_TryParseIntFails_ThrowsException()
{
    // Arrange
    string link =
"https://docs.google.com/spreadsheets/d/1xEbVjwSOi13Z_JS38Q9jjgRI2ePXr3xI4Ua10pixBS8/edit?usp=sharing";

    // Act
    var results = SheetsFacade.GetExamResults(link);
}
```

Selma Ličina 19148

TC4:



Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19109	3	2$	5$

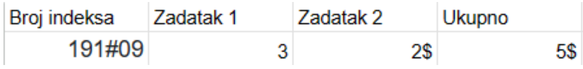
```
[TestMethod]
[ExpectedException(typeof(Exception))]

 public void GetExamResults_TryParseDoubleFails_ThrowsException()
 {
     // Arrange
     string link = "https://docs.google.com/spreadsheets/d/1N85h-
c4LYezp8gF0opxSqGE9Qp-HC6xIskXuJrPn8kQ/edit?usp=sharing";


     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
}
```

*Conditional Coverage

1. Test Case 1 (Both conditions are true):
    - kvp.Key is a valid integer.
    - kvp.Value is a valid double.
2. Test Case 2 (First condition is false):
    - kvp.Key is not a valid integer.
    - kvp.Value can be any value.
3. Test Case 3 (Second condition is false):
    - kvp.Key is a valid integer.
    - kvp.Value is not a valid double.
4. Test Case 4 (Both conditions are false):
    - kvp.Key is not a valid integer.
    - kvp.Value is not a valid double.

Selma Ličina 19148

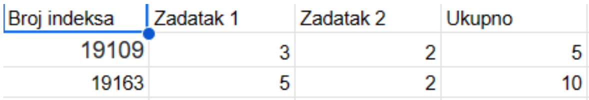
TC1:



Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
191#09	3	2$	5$

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void GetExamResults_BothTryParseFail_ThrowsException()
{
    // Arrange
    string link = "
https://docs.google.com/spreadsheets/d/1cxJwq1Zp0MhPKsNVBEBgORfIINoZQaiJf8PdTt7bBgQ/
edit?usp=sharing";


    // Act
    var results = SheetsFacade.GetExamResults(link);
}
```

TC2:

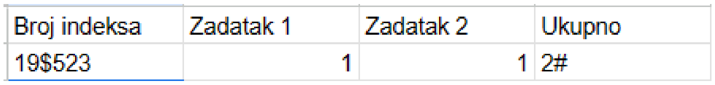


Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19109	3	2	5
19163	5	2	10

```
[TestMethod]
public void GetExamResults_ValidNumbers_ReturnsResults()
{
    // Arrange
    string link =
"https://docs.google.com/spreadsheets/d/1vErXCyteFukR0smnSF8jN7nVhZznzaz2tNJo8pouWjU
/edit#gid=0";

    // Act
    var results = SheetsFacade.GetExamResults(link);

    // Assert
    Assert.IsNotNull(results);
    Assert.AreEqual(2, results.Count);
    Assert.IsTrue(results.ContainsKey(19109));
    Assert.IsTrue(results.ContainsKey(19163));
    Assert.AreEqual(10, results[19163]);
    Assert.AreEqual(5, results[19109]);
}
```

Selma Ličina 19148

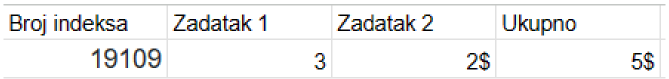
TC3:



Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19$523	1	1	2#

```
[TestMethod]
[ExpectedException(typeof(Exception))]
public void GetExamResults_TryParseIntFails_ThrowsException()
{
    // Arrange
    string link = 
"https://docs.google.com/spreadsheets/d/1xEbVjwSOi13Z_JS38Q9jjgRI2ePXr3xI4Ua10pixBS8
/edit?usp=sharing";

    // Act
    var results = SheetsFacade.GetExamResults(link);
}
```

TC4:



Broj indeksa	Zadatak 1	Zadatak 2	Ukupno
19109	3	2$	5$

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
 public void GetExamResults_TryParseDoubleFails_ThrowsException()
 {
     // Arrange
     string link = "https://docs.google.com/spreadsheets/d/1N85h-
c4LYezp8gF0opxSqGE9Qp-HC6xIskXuJrPn8kQ/edit?usp=sharing";


     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert

 }
```

Selma Ličina 19148

```
*Loop coverage

[TestMethod]
 public void GetExamResults_0Loop()
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/1fjE7ttXCaAPOlcrLzbNur_K_XWEKq_uwrzg00LScOYU
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

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

 [TestMethod]
 public void GetExamResults_1Loop()
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/14fuJGGKvzg2fL3l_bwQnPE_YgGgJTcedcICwBl4xwgk
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
     Assert.IsNotNull(results);
     Assert.AreEqual(1, results.Count);
 }

 [TestMethod]
 public void GetExamResults_2Loop()
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/1vErXCyteFukR0smnSF8jN7nVhZznzaz2tNJo8pouWjU
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
     Assert.IsNotNull(results);
     Assert.AreEqual(2, results.Count);
 }
```

Selma Ličina 19148

```
[TestMethod]
 public void GetExamResults_RandomNumberLoop() //validStringResults.Count = 8
 {
     // Arrange
     string link = "https://docs.google.com/spreadsheets/d/1Iy0NTNuxLLV4B-
QLD1sqv5z5fRqcFtmxMwwcTtNZ2iI/edit?usp=sharing";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
     Assert.IsNotNull(results);
     Assert.AreEqual(8, results.Count);
 }

[TestMethod]
 public void GetExamResults_NNMinusNPlusLoop() //validStringResults.Count = 1,0,2
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/14fuJGGKvzg2fL3l_bwQnPE_YgGgJTcedcICwBl4xwgk
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
     Assert.IsNotNull(results);
     Assert.AreEqual(1, results.Count);
 }
```

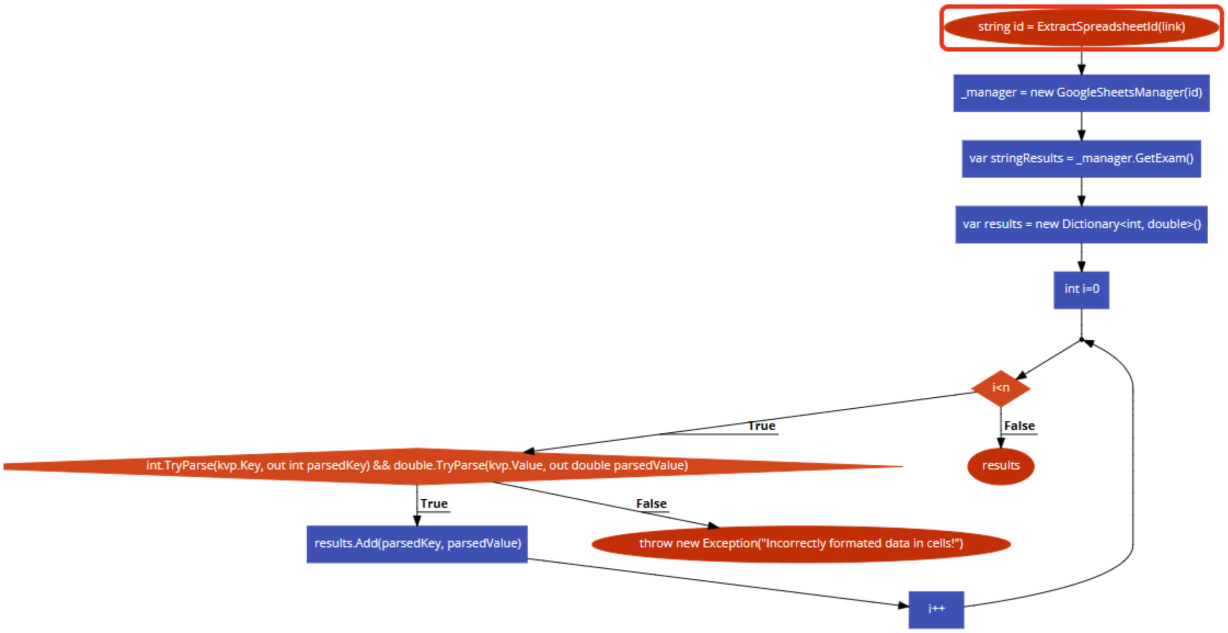

string id = ExtractSpreadsheetId(link)
_manager = new GoogleSheetsManager(id)
var stringResults = _manager.GetExam()
var results = new Dictionary<int, double>()
int i=0
i<n
True
False
results
int.TryParse(kvp.Key, out int parsedKey) && double.TryParse(kvp.Value, out double parsedValue)
True
False
results.Add(parsedKey, parsedValue)
throw new Exception("Incorrectly formated data in cells!")
i++

Figure 1 Dijagram toka modula

```
i = stringResults[i];
n = stringResults.Count
```

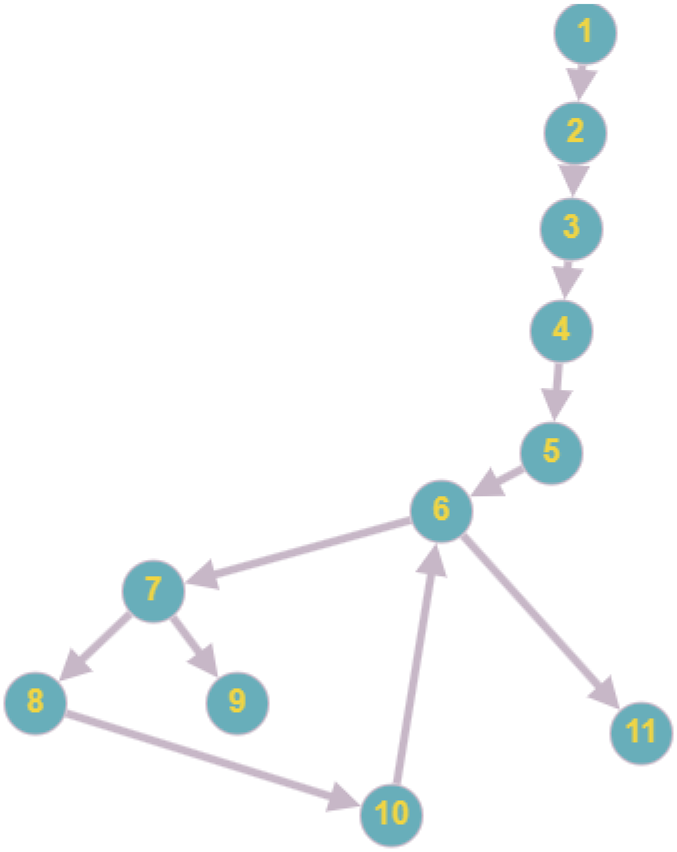


Figure 2 Programski graf modula

Broj puta	Put
1	1-2-3-4-5-6-11
2	1-2-3-4-5-6-7-8-10-6-11
3	1-2-3-4-5-6-7-9
4	1-2-3-4-5-6-7-8-10-6-7-9
5	1-2-3-4-5-6-7-8-10-6-7-8-10-6-7-9
…	…

Minimalno 5 testna slučaja, broj slučajeva se povećava s obzirom na stringResults dužinu.

Plavom bojom je označen maksimalan skup neovisnih puteva.

Selma Ličina 19148

```
TC1:
[TestMethod]
 public void GetExamResults_Path1_ReturnsValidResults()
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/14fuJGGKvzg2fL3l_bwQnPE_YgGgJTcedcICwBl4xwgk
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
     Assert.IsNotNull(results);
     Assert.AreEqual(1, results.Count);
 }

TC2:
 [TestMethod]
[ExpectedException(typeof(ArgumentException))]
 public void GetExamResults_Path2_ThrowsException()
 {
     // Arrange
     string link = 
"https://docs.google.com/spreadsheets/d/14fuJGGKvzg2fL3l_bwQnPE_YgGgJTcedcICwBl4xwgk
/edit#gid=0";

     // Act
     var results = SheetsFacade.GetExamResults(link);

     // Assert
}
```