

**گزارش پروژه دوم**

**نام و نام خانوادگی**

نگار فتحی

**شماره دانشجویی**

97723137

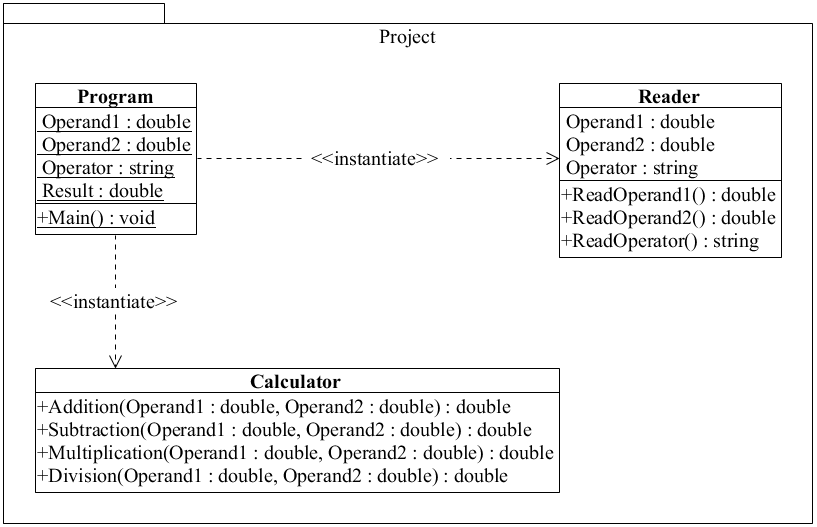
**استاد**

دکتر سعید پارسا

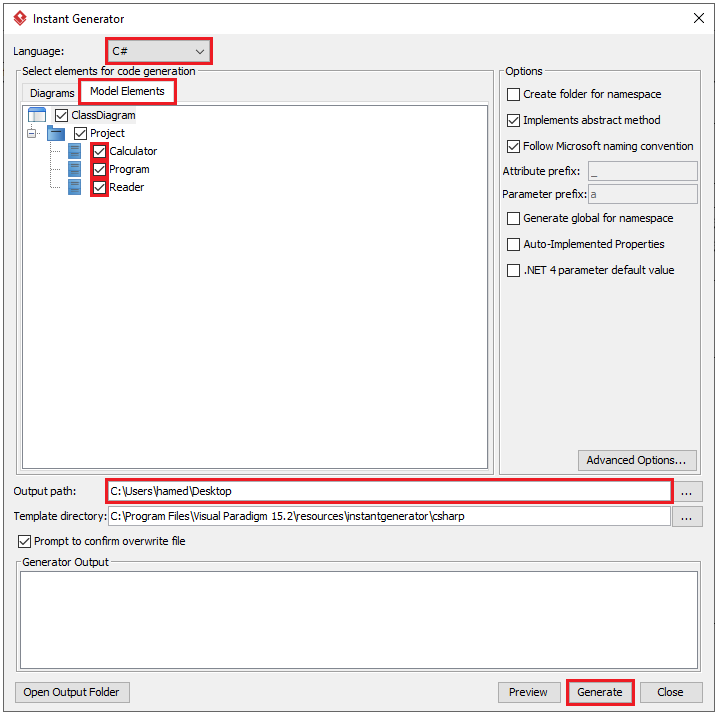
**درس**

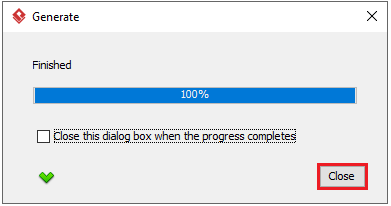
کامپایلر پیشرفته

در پروژه انجام‏شده، بعداز رسم نمودار کلاس در محیط Visual Paradigm، تولید کد C# از روی نمودار کلاس، ایجاد پروژه در محیط Visual Studio، افزودن کلاس‏ها به پروژه و درنهایت پیاده‏سازی کلاس‏ها، متدهای برنامه به‏کمک ابزار NUnit تحت آزمون واحد قرار گرفتند.

* **رسم نمودار کلاس در محیط Visual Paradigm:**
* **تولید کد C# از روی نمودار کلاس:**

Visual Paradigm ⇒ Code ⇒ Instant Generator ⇒





کد تولید‏شده برای کلاس Program:

|  |
| --- |
| using System;  namespace Project {  public class Program {  private static double operand1;  private static double operand2;  private static string operator;  private static double result;  public static void Main() {  throw new System.Exception("Not implemented");  }  }  } |

کد تولید‏شده برای کلاس Reader:

|  |
| --- |
| using System;  namespace Project {  public class Reader {  private double operand1;  private double operand2;  private string operator;  public double ReadOperand1() {  throw new System.Exception("Not implemented");  }  public double ReadOperand2() {  throw new System.Exception("Not implemented");  }  public string ReadOperator() {  throw new System.Exception("Not implemented");  }  }  } |

کد تولید‏شده برای کلاس Calculator:

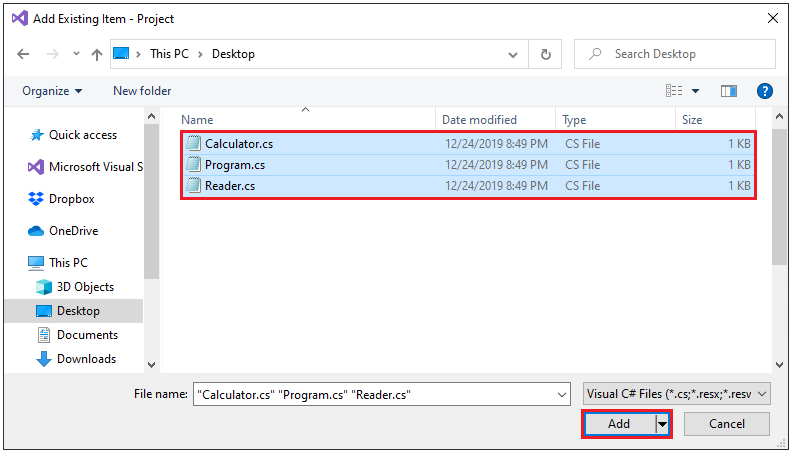
|  |
| --- |
| using System;  namespace Project {  public class Calculator {  public double Addition(double operand1, double operand2) {  throw new System.Exception("Not implemented");  }  public double Subtraction(double operand1, double operand2) {  throw new System.Exception("Not implemented");  }  public double Multiplication(double operand1, double operand2) {  throw new System.Exception("Not implemented");  }  public double Division(double operand1, double operand2) {  throw new System.Exception("Not implemented");  }  }  } |

* **ایجاد پروژه در محیط Visual Studio:**

Visual Studio ⇒ File ⇒ New ⇒ Project ⇒ Visual C# ⇒ Empty Project (.NET Framework) ⇒ Neme: Project ⇒ Framework: .NET Framework 4.6.1 ⇒ Ok.

* **افزودن کلاس‏ها به پروژه:**

Visual Studio ⇒ Solution Explorer ⇒ Project ⇒ Add ⇒ Existing Item... ⇒

****

* **پیاده‏سازی کلاس‏ها:**

پیاده‏سازی کلاس Program:

|  |
| --- |
| using System;  namespace Project  {  public class Program  {  private static double operand1;  private static double operand2;  private static string Operator;  private static double result;  public static void Main()  {  **Reader reader = new Reader();**  **operand1 = reader.ReadOperand1();**  **operand2 = reader.ReadOperand2();**  **Operator = reader.ReadOperator();**  **Calculator calculator = new Calculator();**  **switch (Operator)**  **{**  **case "+":**  **result = calculator.Addition(operand1, operand2);**  **break;**  **case "-":**  **result = calculator.Subtraction(operand1, operand2);**  **break;**  **case "\*":**  **result = calculator.Multiplication(operand1, operand2);**  **break;**  **case "/":**  **result = calculator.Division(operand1, operand2);**  **break;**  **}**  **Console.Write("Result: " + result);**  **Console.ReadLine();**  }  }  } |

پیاده‏سازی کلاس Reader:

|  |
| --- |
| using System;  namespace Project  {  public class Reader  {  private double operand1;  private double operand2;  private string Operator;  public double ReadOperand1()  {  **Console.Write("Operand1: ");**  **operand1 = double.Parse(Console.ReadLine());**  **return operand1;**  }  public double ReadOperand2()  {  **Console.Write("Operand2: ");**  **operand2 = double.Parse(Console.ReadLine());**  **return operand2;**  }  public string ReadOperator()  {  **Console.Write("Operator: ");**  **Operator = Console.ReadLine();**  **return Operator;**  }  }  } |

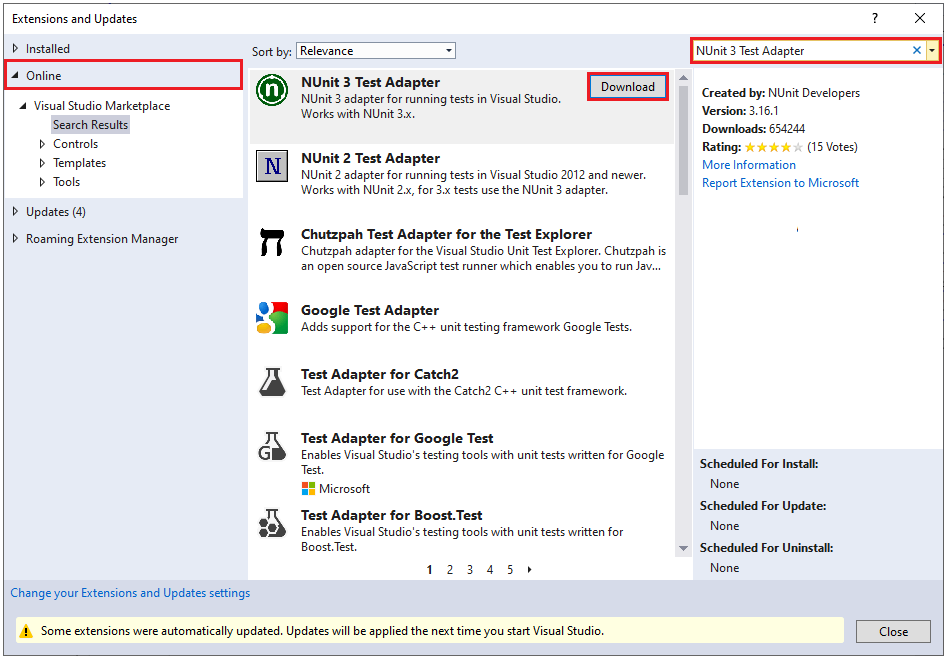
پیاده‏سازی کلاس Calculator:

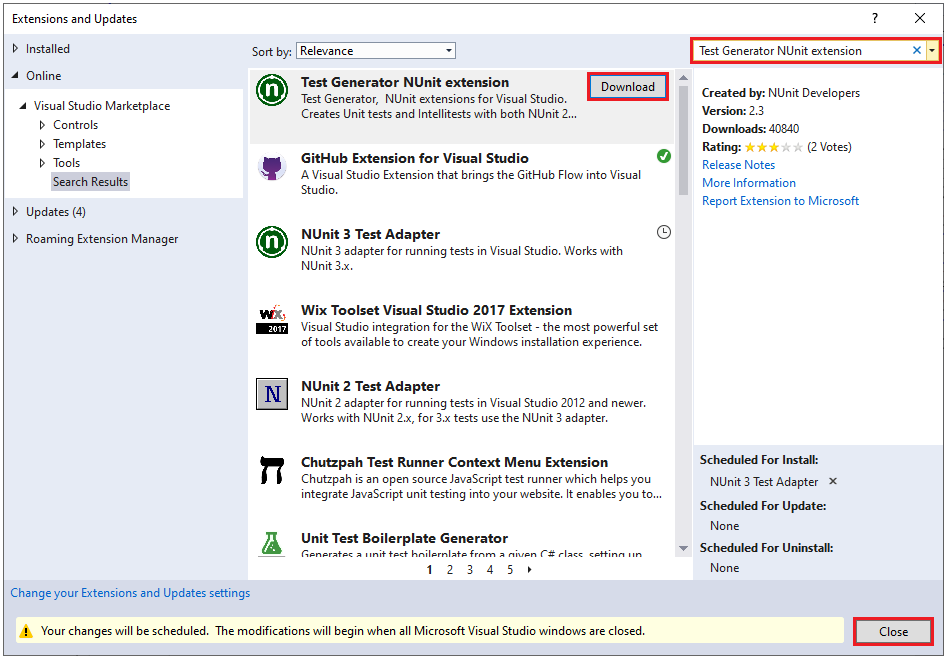
|  |
| --- |
| using System;  namespace Project  {  public class Calculator  {  public double Addition(double operand1, double operand2)  {  **return operand1 + operand2;**  }  public double Subtraction(double operand1, double operand2)  {  **return operand1 - operand2;**  }  public double Multiplication(double operand1, double operand2)  {  **return operand1 \* operand2;**  }  public double Division(double operand1, double operand2)  {  **return operand1 / operand2;**  }  }  } |

* **نصب NUnit:**

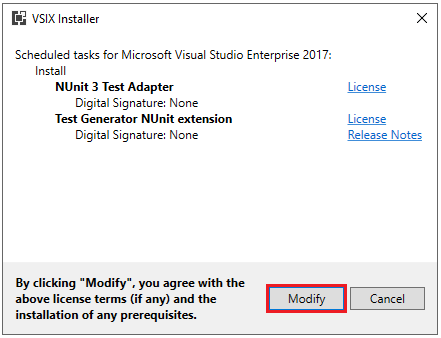
ازآن‏جایی‏که قرار است متدهای برنامه موردنظر را به‏کمک ابزار NUnit تحت آزمون واحد قرار دهیم، باید مراحل نصب آن به‏صورت زیر طی شود.

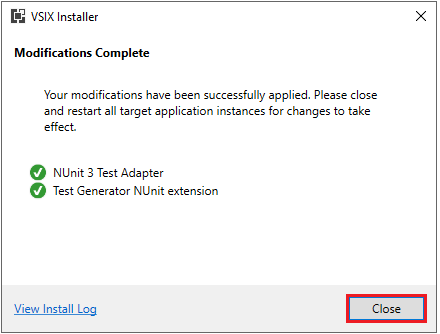
Visual Studio ⇒ Tools ⇒ Extension and Updates… ⇒





⇒ Close Visual Studio ⇒

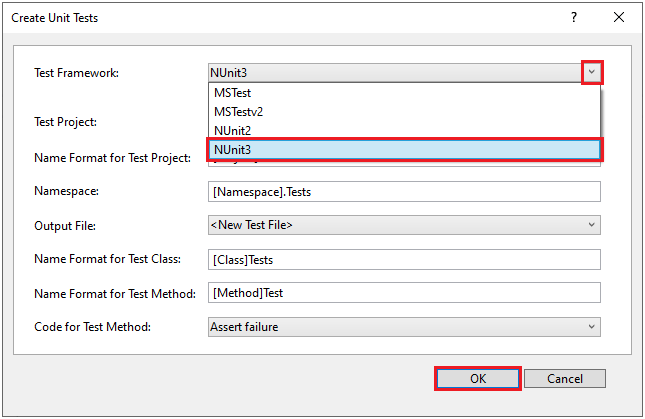




⇒ Open Visual Studio.

* **آزمون واحد:**

در مرحله اول، قصد داریم متدهای Addition، Subtraction، Multiplication و Division کلاس Calculator را تحت آزمون واحد قرار دهیم. بدین‏منظور به‏ترتیب بر روی این متدها کلیک راست کرده و بعد از زدن گزینه Create Unit Tests مطابق شکل زیر عمل می‏کنیم.



حال در قسمت Solution Explorer پوشه‏ای به نام ProjectTests ایجاد شده‏است که حاوی یک کلاس با نام CalculatorTests.cs است. محتویات این کلاس به‏صورت زیر است.

|  |
| --- |
| using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class CalculatorTests  {  [Test()]  public void AdditionTest()  {  Assert.Fail();  }  [Test()]  public void SubtractionTest()  {  Assert.Fail();  }  [Test()]  public void MultiplicationTest()  {  Assert.Fail();  }  [Test()]  public void DivisionTest()  {  Assert.Fail();  }  }  } |

به‏منظور این‏که متدهای Addition، Subtraction، Multiplication و Division کلاس Calculator را تحت آزمون واحد قرار دهیم، متدهای AdditionTest، SubtractionTest، MultiplicationTest و DivisionTest کلاس CalculatorTests.cs را به‏صورت زیر بازنویسی می‏کنیم.

|  |
| --- |
| using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class CalculatorTests  {  [Test()]  public void AdditionTest()  {  **Calculator calculator = new Calculator();**  **double result = calculator.Addition(1, 2);**  **Assert.AreEqual(3, result);**  }  [Test()]  public void SubtractionTest()  {  **Calculator calculator = new Calculator();**  **double result = calculator.Subtraction(1, 2);**  **Assert.AreEqual(-1, result);**  }  [Test()]  public void MultiplicationTest()  {  **Calculator calculator = new Calculator();**  **double result = calculator.Multiplication(1, 2);**  **Assert.AreEqual(2, result);**  }  [Test()]  public void DivisionTest()  {  **Calculator calculator = new Calculator();**  **double result = calculator.Division(1, 2);**  **Assert.AreEqual(0.5, result);**  }  }  } |

در مرحله دوم، قصد داریم متدهای ReadOperand1، ReadOperand2 و ReadOperator کلاس Reader را تحت آزمون واحد قرار دهیم. بدین‏منظور مطابق آنچه برای متدهای کلاس Calculator گفته شد، عمل می‏کنیم. سپس در پوشه‏ ProjectTests در قسمت Solution Explorer شاهد ایجاد کلاس دیگری با نام ReaderTests.cs خواهیم بود که محتویات آن به‏صورت زیر است.

|  |
| --- |
| using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class ReaderTests  {  [Test()]  public void ReadOperand1Test()  {  Assert.Fail();  }  [Test()]  public void ReadOperand2Test()  {  Assert.Fail();  }  [Test()]  public void ReadOperatorTest()  {  Assert.Fail();  }  }  } |

این کلاس را به‏صورت زیر بازنویسی می‏کنیم.

|  |
| --- |
| **using System;**  **using System.IO;**  using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class ReaderTests  {  [Test()]  public void ReadOperand1Test()  {  **var input = new StringReader("1");**  **Console.SetIn(input);**  **Reader reader = new Reader();**  **double result = reader.ReadOperand1();**  **Assert.AreEqual(1, result);**  }  [Test()]  public void ReadOperand2Test()  {  **var input = new StringReader("2");**  **Console.SetIn(input);**  **Reader reader = new Reader();**  **double result = reader.ReadOperand2();**  **Assert.AreEqual(2, result);**  }  [Test()]  public void ReadOperatorTest()  {  **var input = new StringReader("+");**  **Console.SetIn(input);**  **Reader reader = new Reader();**  **string result = reader.ReadOperator();**  **Assert.AreEqual("+", result);**  }  }  } |

حال قصد داریم متد Main کلاس Program را تحت آزمون واحد قرار دهیم. اما با‏توجه‏به نحوه پیاده‏سازی صورت‏گرفته، این متد قابلیت آزمون ندارد. بدین‏منظور پیاده‏سازی را به صورت زیر تغییر می‏دهیم.

پیاده‏سازی جدید کلاس Program:

|  |
| --- |
| using System;  namespace Project  {  public class MyClass  {  private IReader iReader;  private ICalculator iCalculator;  private static double operand1;  private static double operand2;  private static string Operator;  private static double result;  public MyClass(IReader iReader, ICalculator iCalculator)  {  this.iReader = iReader;  this.iCalculator = iCalculator;  }  public void MyMethod()  {  operand1 = iReader.ReadOperand1();  operand2 = iReader.ReadOperand2();  Operator = iReader.ReadOperator();  switch (Operator)  {  case "+":  result = iCalculator.Addition(operand1, operand2);  break;  case "-":  result = iCalculator.Subtraction(operand1, operand2);  break;  case "\*":  result = iCalculator.Multiplication(operand1, operand2);  break;  case "/":  result = iCalculator.Division(operand1, operand2);  break;  }  Console.Write("Result: " + result);  Console.ReadLine();  }  }  public class Program  {  public static void Main()  {  Reader reader = new Reader();  Calculator calculator = new Calculator();  MyClass myClass = new MyClass(reader, calculator);  myClass.MyMethod();  }  }  } |

پیاده‏سازی جدید کلاس Reader:

|  |
| --- |
| using System;  namespace Project  {  public interface IReader  {  double ReadOperand1();  double ReadOperand2();  string ReadOperator();  }  **public class ReaderStub : IReader**  **{**  **public double ReadOperand1()**  **{**  **return 1;**  **}**  **public double ReadOperand2()**  **{**  **return 2;**  **}**  **public string ReadOperator()**  **{**  **return "+";**  **}**  **}**  public class Reader : IReader  {  private double operand1;  private double operand2;  private string Operator;  public double ReadOperand1()  {  Console.Write("Operand1: ");  operand1 = double.Parse(Console.ReadLine());  return operand1;  }  public double ReadOperand2()  {  Console.Write("Operand2: ");  operand2 = double.Parse(Console.ReadLine());  return operand2;  }  public string ReadOperator()  {  Console.Write("Operator: ");  Operator = Console.ReadLine();  return Operator;  }  }  } |

در پیاده‏سازی جدید کلاس Reader، کلاسی به نام ReaderStub اضافه شده‏است که برای انجام آزمون واحد متدی که درون خود متدهای کلاس Reader را فراخوانی می‏کند، ضروری است.

پیاده‏سازی جدید کلاس Calculator:

|  |
| --- |
| using System;  namespace Project  {  public interface ICalculator  {  double Addition(double operand1, double operand2);  double Subtraction(double operand1, double operand2);  double Multiplication(double operand1, double operand2);  double Division(double operand1, double operand2);  }  **public class CalculatorStub : ICalculator**  **{**  **public double Addition(double operand1, double operand2)**  **{**  **return 3;**  **}**  **public double Subtraction(double operand1, double operand2)**  **{**  **return -1;**  **}**  **public double Multiplication(double operand1, double operand2)**  **{**  **return 2;**  **}**  **public double Division(double operand1, double operand2)**  **{**  **return 0.5;**  **}**  **}**  public class Calculator : ICalculator  {  public double Addition(double operand1, double operand2)  {  return operand1 + operand2;  }  public double Subtraction(double operand1, double operand2)  {  return operand1 - operand2;  }  public double Multiplication(double operand1, double operand2)  {  return operand1 \* operand2;  }  public double Division(double operand1, double operand2)  {  return operand1 / operand2;  }  }  } |

در پیاده‏سازی جدید کلاس Calculator، کلاسی به نام CalculatorStub اضافه شده‏است که برای انجام آزمون واحد متدی که درون خود متدهای کلاس Calculator را فراخوانی می‏کند، ضروری است.

اکنون می‏توان متد MyMethod کلاس MyClass را تحت آزمون واحد قرار داد. بدین‏منظور مطابق آنچه برای متدهای کلاس Calculator گفته شد، عمل می‏کنیم. سپس در پوشه‏ ProjectTests در قسمت Solution Explorer شاهد ایجاد کلاس دیگری با نام MyClassTests.cs خواهیم بود که محتویات آن به‏صورت زیر است.

|  |
| --- |
| using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class MyClassTests  {  [Test()]  public void MyMethodTest()  {  Assert.Fail();  }  }  } |

این کلاس را به‏صورت زیر بازنویسی می‏کنیم.

|  |
| --- |
| **using System;**  **using System.IO;**  using NUnit.Framework;  namespace Project.Tests  {  [TestFixture()]  public class MyClassTests  {  [Test()]  public void MyMethodTest()  {  **var output = new StringWriter();**  **Console.SetOut(output);**  **var input = new StringReader("0");**  **Console.SetIn(input);**  **ReaderStub readerStub = new ReaderStub();**  **CalculatorStub calculatorStub = new CalculatorStub();**  **MyClass myClass = new MyClass(readerStub, calculatorStub);**  **myClass.MyMethod();**  **Assert.That(output.ToString(), Is.EqualTo("Result: 3"));**  }  }  } |

حال تمامی آزمون‏های نوشته‏شده را اجرا می‏کنیم. بدین‏منظور باید مراحل زیر طی شود.

Visual Studio ⇒ Test ⇒ Run ⇒ All Tests.

می‏توان نتایج آزمون واحد برای هریک از متدها را در بخش Test Explorer مشاهده کرد. برای مثال درمورد مثال‏های موجود در این گزارش نتایج به‏صورت زیر است.

