# Programming Advanced for QA – Regular Exam 17 August 2025

**Submit your zip file here:**

<https://alpha.judge.softuni.org/contests/programming-advanced-for-qa-regular-exam-17-august-2025/5148>

# 1. Unit Test: Digits and Symbols Counter

Test a given method which **takes a string input representing a whole sentence** and **checks each character to count the number of even digits, odd digits, and non-digit symbols**:

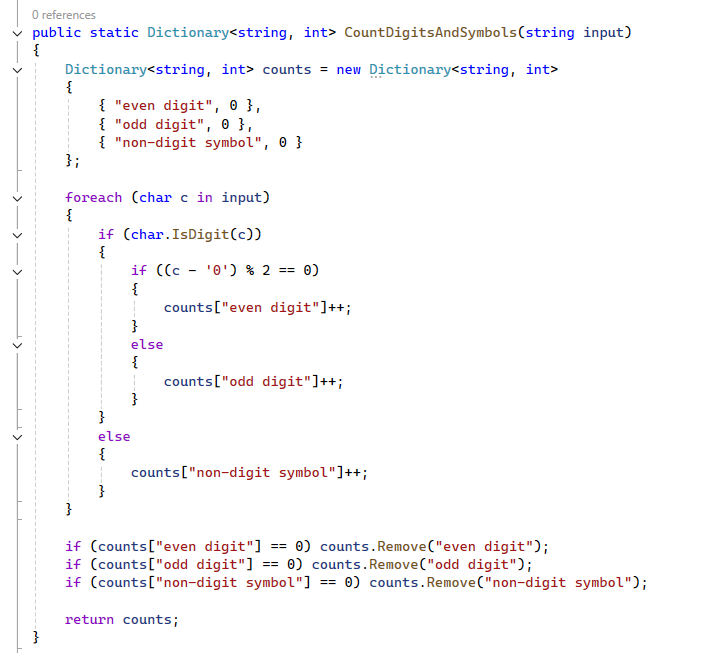
* If a **character is a digit** and **is even (0, 2, 4, 6, 8)**, it is classified as **even digit**.
* If a **character is a digit** and **is odd (1, 3, 5, 7, 9)**, it is classified as **odd digit**.
* Any **non-digit characters (letters, punctuation, symbols, spaces, etc.)** are classified as **non-digit symbols**.

**IMPORTANT:** White spaces, punctuation, and all non-numeric characters will be counted as non-digit symbols.

### Examples

|  |  |
| --- | --- |
| **Argument** | **Returned dictionary** |
| (empty string) | (empty dictionary) |
| Hello World | non-digit symbol -> 11 times |
| 2468XYZ | even digit -> 4 times  non-digit symbols -> 3 times |
| QA is cool 579? | odd digit -> 3 times  non-digit symbols -> 12 times |
| 123abc! | even digit -> 1 time  odd digit -> 2 times  non-digit symbols -> 4 times |

The method is found in the DigitAndSymbolCounter.cs file:

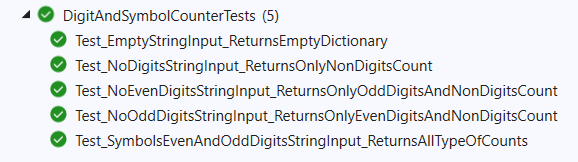


You are given a **test** **file** DigitAndSymbolCounterTests.cs containing **5 empty tests**. Implement all tests:



**Note! You may need to test if result collection is equal to expected.**

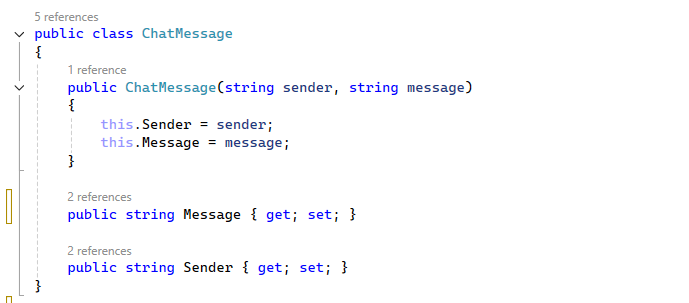
When you are ready make sure your **tests run:**



**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

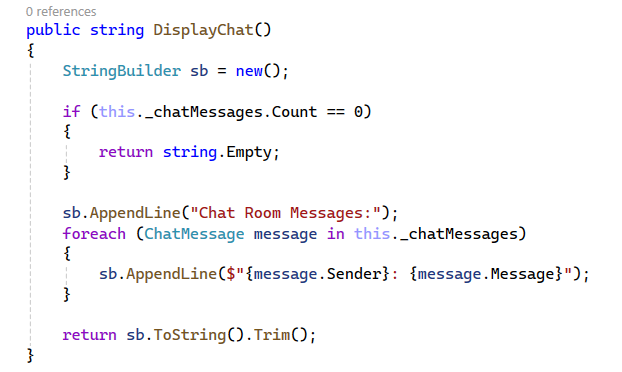
# Unit Test: Chat

You are given a **folder of 2 classes -** ChatRoomandChatMessage. The ChatMessage **class** is just a helper class:



The ChatRoomclass holds a **list** and **methods** for **using the collection** that you will **test**:

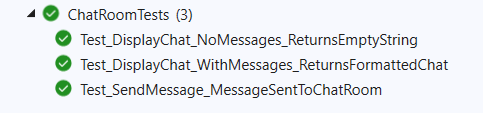




You will need to use the test file ChatRoomTests.cs, inside they are **3 empty tests with a setup method:**



When you are ready make sure your **tests run:**



**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

**03. JS Unit Testing: Workforce Management**

After opening the project, run **npm install** in the console.

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **workforceManagement**, which represents an object. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });  *//* ***TODO:*** …  }); |

The object should have the following functionality:

**recruitStaff (name, role,** **experience) -** A function that accepts three parameters: **string**, **string**, and **number**.

* If the value of the string **role** is different from "**Developer**", **throw** an error: **`We are not currently hiring for this role.`**
* To be hired, the **employee** must meet the **following requirements**:
  + If the **experience** is **greater** than or **equal** to **4**, **return** the string:

**`{name} has been successfully recruited for the role of {role}.`**

* Otherwise, if the above conditions are not met, **return** the following message:

**`{name} is not suitable for this role.`**

* There is **no** need for **validation** for the **input**, you will always be given a string, string, and number.
* **computeWages (hoursWorked) -** A function that accepts one parameter: **number**.
* Workers in this company receive **equal** pay per **hour** and this is **BGN 18**.
* You need to **calculate** the salary by **multiplying** the pay **for one hour** by the number of **hours**.
* **Also**, if the employee has been working for **more than 160 hours**, he must receive an additional **BGN 1500 bonus.**
* Finally, **return** the employee's salary.
* You need to validate the input, if the **hoursWorked** are not a **number**, or are a **negative** number, **throw** an error: "**Invalid hours**".
* **dismissEmployee (workforce, employeeIndex) -** A function that accepts an array and number.
  + The **workforce** array will store the names of its employees (["**Petar**", "**Ivan**", "**George**"…]).
  + You must **remove** an **element** (employee) from the **array** that is located on the **employeeIndex** specified as a parameter.
* Finally, **return** the changed array of workforce as a string, **joined** by a **comma** and a **space**.
  + There is a need for validation for the input, an **array** and index may not always be valid. In case of submitted **invalid** parameters, **throw** an error "**Invalid input**":
    - If passed **workforce** parameteris not an array.
    - If the **employeeIndex** is not a number and is outside the limits of the array.

**JS Code**

To ease you in the process, you are provided with an implementation that meets all of the specification requirements for the **workforceManagement** object:

**Submission**

Submit your tests inside a **describe()** statement, as shown above.