ΠΕΡΙΕΧΟΜΕΝΑ

ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ

Στο παρόν έγγραφο αναλύεται η υλοποίηση της απαλλακτικής εργασίας (Μέρος β) για το μάθημα της Ποιότητας και Αξιοπιστίας Λογισμικού. Ο σκοπός της είναι η ανάπτυξη μιας βιβλιοθήκης .dll, η οποία περιέχει μια σειρά από μεθόδους και συναρτήσεις για την διαχείριση του τμήματος προσωπικού μιας εταιρείας.

ΤΕΧΝΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε το Microsoft Visual Studio 2022 το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) από τη Microsoft και το framework .NET 4.8.

ΜΕΘΟΔΟΙ & ΣΥΝΑΡΤΗΣΕΙΣ

1. bool ValidName(string Name)

Η συνάρτηση αυτή δέχεται ως παράμετρο ένα κείμενο και επιστρέφει τιμή true ή false αν η παράμετρος μπορεί να αντιστοιχεί σε ονοματεπώνυμο υπαλλήλου ή όχι, αντίστοιχα.

Α. Κώδικας

```
public static bool ValidName(string name)
    {
        string[] parts = name.Split(' ');

        // Όνομα + Επώνυμο
        if (parts.Length < 2)
        {
            return false;
        }

        // Πρέπει να ξεκινάει με κεφαλαίο
        if (char.IsLower(name[0]))
        {
            return false;
        }

        for (int i = 0; i < name.Length; i++)
        {
            char c = name[i];

            // Χαρακτήρας που δεν είναι γράμμα, ` `, ή `-`
            if (!char.IsLetter(c) && c != '-' && c != ' ')
            {
                return false;
            }

            // Το `-` δεν μπορεί να είναι στην άκρη του μηνύματος
            if (c == '-' && (i == 0 || i + 1 == name.Length))
            {
                return false;
            }
        }

        return true;
    }
```

Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	Δοκιμαστικά Δεδομένα Εισόδου	Αναμενόμενα Αποτελέσματα
#id	name	flag
1	Stavros Nastoulis	TRUE
2	Stavros-Thomas Nastoulis	TRUE
3	John-Doe	FALSE
4	JohN	FALSE
5	george brown	FALSE
6	First-Name Middle-Name Last-Name	TRUE
7	St@vros N@stouli	FALSE
8	Savros1 Nast	FALSE
9	-Savros Nast	FALSE
10	Savros Nast-	FALSE

Γ. Unit Tests

```
public void TestValidName()
      {
         // Δημιουργία Test Cases
         object[,] testcases = {
            {1, "Stavros Nastoulis", true},
            {2, "Stavros-Thomas Nastoulis", true},
            {3, "John-Doe", false},
            {4, "JohN", false},
            {5, "george brown", false},
            {6, "First-Name Middle-Name Last-Name", true},
            {7, "St@vros N@stouli", false},
            {8, "Savros1 Nast", false},
            {9, "-Savros Nast", false},
            {10, "Savros Nast-", false}
         };

         bool failed = false;

         for (int i = 0; i < testcases.GetLength(0); i++)
         {
            bool res = HRLib.Hr.ValidName((string)testcases[i, 1]);

            try
            {
               Assert.AreEqual((bool)testcases[i, 2], res);
            }
            catch (Exception e)
            {
               //Απέτυχε η περίπτωση
               failed = true;
               //Καταγράφουμε την περίπτωση ελέγχου που
               Console.WriteLine("Failed Test Case: {0}. Reason: {1}.", (int)testcases[i, 0], e.Message);
            };
         }

         //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
         if (failed) Assert.Fail();
      }
```

Δ. Αναφορές Ελέγχου

Αναφορά σφάλματος #1

Failed Test Case: 3. Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>. .

1. Αναγνωριστικό Σφάλματος (αποτελείται από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)

Failed Test Case: 3

2. Όνομα μεθόδου ελέγχου (test method) που απέτυχε

TestValidName

3. Περιγραφή ελέγχου που πραγματοποιήθηκε

Αναμενόμενη τιμή από τον χρήστη: False

Τιμή που έβγαλε το πρόγραμμα: True

4. Αναφορά σφάλματος που παρουσιάστηκε

Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.

Αναφορά σφάλματος #2

Failed Test Case: 4. Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>. .

1. Αναγνωριστικό Σφάλματος (αποτελείται από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)

Failed Test Case: 4

2. Όνομα μεθόδου ελέγχου (test method) που απέτυχε

TestValidName

3. Περιγραφή ελέγχου που πραγματοποιήθηκε

Αναμενόμενη τιμή από τον χρήστη: False

Τιμή που έβγαλε το πρόγραμμα: True

4. Αναφορά σφάλματος που παρουσιάστηκε

Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.

Αναφορά σφάλματος #3

Failed Test Case: 5. Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>. .

1. Αναγνωριστικό Σφάλματος (αποτελείται από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)

Failed Test Case: 5

2. Όνομα μεθόδου ελέγχου (test method) που απέτυχε

TestValidName

3. Περιγραφή ελέγχου που πραγματοποιήθηκε

Αναμενόμενη τιμή από τον χρήστη: False

Τιμή που έβγαλε το πρόγραμμα: True

4. Αναφορά σφάλματος που παρουσιάστηκε

Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.

Αναφορά σφάλματος #4

Failed Test Case: 7. Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>. .

1. Αναγνωριστικό Σφάλματος (αποτελείται από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)

Failed Test Case: 7

2. Όνομα μεθόδου ελέγχου (test method) που απέτυχε

TestValidName

3. Περιγραφή ελέγχου που πραγματοποιήθηκε

Αναμενόμενη τιμή από τον χρήστη: False

Τιμή που έβγαλε το πρόγραμμα: True

4. Αναφορά σφάλματος που παρουσιάστηκε

Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.

Αναφορά σφάλματος #5
Failed Test Case: 8. Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>. .

1. Αναγνωριστικό Σφάλματος (αποτελείται από ένα λεκτικό αναγνωριστικό του σφάλματος και έναν αύξοντα αριθμό)

Failed Test Case: 8

2. Όνομα μεθόδου ελέγχου (test method) που απέτυχε

TestValidName

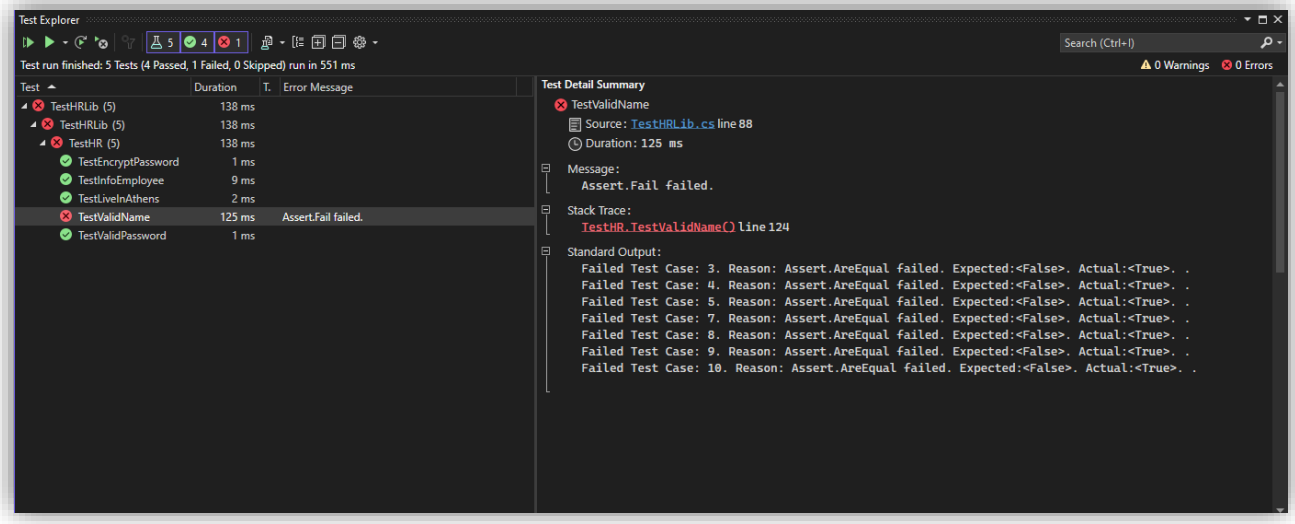
3. Περιγραφή ελέγχου που πραγματοποιήθηκε

Αναμενόμενη τιμή από τον χρήστη: False

Τιμή που έβγαλε το πρόγραμμα: True

4. Αναφορά σφάλματος που παρουσιάστηκε

Reason: Assert.AreEqual failed. Expected:<False>. Actual:<True>.



2. bool ValidPassword(string Password)

Η συνάρτηση αυτή δέχεται ως παράμετρο ένα κείμενο και επιστρέφει τιμή true ή false αν η παράμετρος αποτελεί έναν αποδεκτό κωδικό πρόσβασης. Προϋποθέσεις κωδικού: α. Τουλάχιστον 12 χαρακτήρες, β. Συνδυασμός κεφαλαίων γραμμάτων, πεζών γραμμάτων, αριθμών και συμβόλων (τουλάχιστον έναν χαρακτήρα από το καθένα), γ. Τα γράμματα να είναι λατινικοί χαρακτήρες, δ. Να ξεκινάει από κεφαλαίο γράμμα και να τελειώνει με αριθμό.

Α. Κώδικας

```
public static bool ValidPassword(string Password)
    {
        bool has_lower = false;
        bool has_symbol = false;

        if (Password.Length <= 12)
        {
            // Πρέπει ο κωδικός να είναι τουλάχιστον 12 χαρακτήρες
            return false;
        }

        foreach (char c in Password)
        {
            if (char.IsLetter(c))
            {
                if (char.IsLower(c)) has_lower = true;

                if (!(c >= 'A' && c <= 'Z') && !(c >= 'a' && c <= 'z')) return false;  // Μόνο Λατινικοί χαρακτήρες
            }

            if (!char.IsLetterOrDigit(c)) has_symbol = true;
        }
        if (!has_lower || !has_symbol) return false; // Πρέπει να έχει τουλάχιστον ένα πεζό και τουλάχιστον ένα σύμβολο

        if (!char.IsUpper(Password, 0) || !char.IsDigit(Password, Password.Length - 1))
        {
            // Πρέπει να ξεκινάει με κεφαλαίο και να τελειώνει με αριθμό
            return false;
        }

        return true;
    }
```

Β. Πίνακας – περιπτώσεις ελέγχου

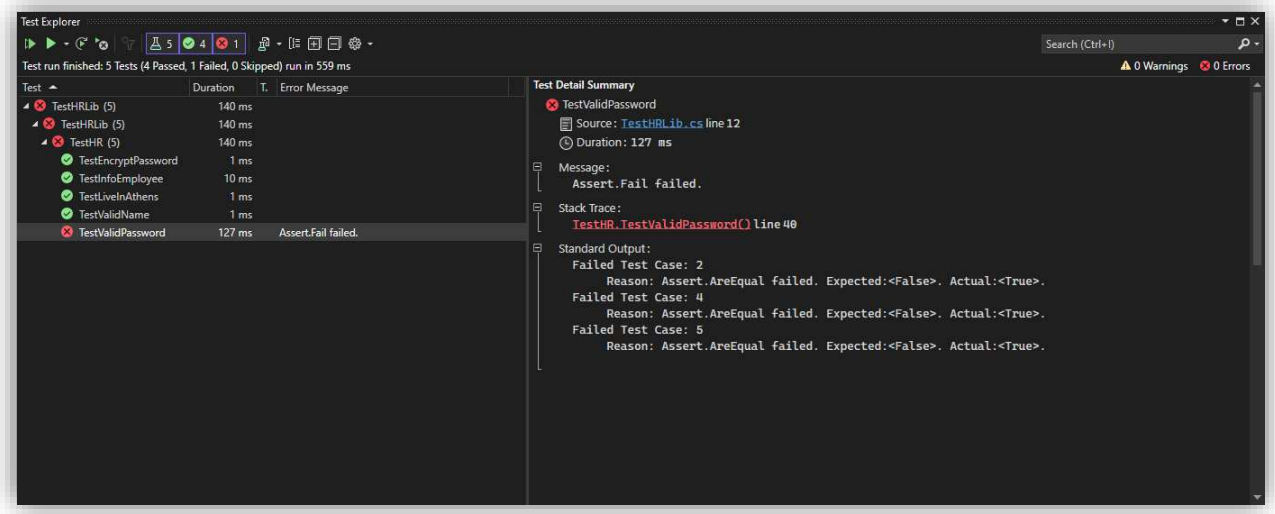
Περ.Ελ.	Δοκιμαστικά Δεδομένα Εισόδου	Αναμενόμενα Αποτελέσματα
#id	password	flag
1	Mix3dP@ssw0rd3	TRUE
2	ShortPW123	FALSE
3	Ge0metricP@ss1	TRUE
4	!SpecialPW123	FALSE
5	NoNumbersHere!	FALSE

Γ. Unit Tests

```
public void TestValidPassword()
    // bool ValidPassword(string Password)
    {
        // Δημιουργία Test Cases
```

```
        object[,] testcases = {
            {1, true, "Mix3dP@ssw0rd3"},
            {2, false, "ShortPW123"},
            {3, true, "Ge0metricP@ss1"},
            {4, false, "!SpecialPW123"},
            {5, false, "NoNumbersHere!"}
        };

        bool failed = false;

        for (int i = 0; i < testcases.GetLength(0); i++)
        {
            try
            {
                // Δίνουμε ως παράμετρους στην Assert.AreEqual τα στοιχεία του test case
                Assert.AreEqual((bool)testcases[i, 1], HRLib.Hr.ValidPassword((string)testcases[i, 2]));
            }
            catch (Exception exception)
            {
                failed = true;
                Console.WriteLine("Failed Test Case: {0}\n \t Reason: {1} ", (int)testcases[i, 0], exception.Message);
            }
        }

        if (failed) Assert.Fail();
    }
```

Δ. Αναφορές Ελέγχου



3. EncryptPassword(string Password, ref string EncryptedPW)

Η μέθοδος αυτή δέχεται ως παράμετρο ένα string που είναι κωδικός και επιστρέφει τον κρυπτογραφημένο κωδικό με βάση τον Κώδικα του Καίσαρα (Caesar's Cipher), με αλφάβητο το ASCII (128 χαρακτήρες) και ολίσθηση κατά 5 θέσεις. Το αποτέλεσμα επιστρέφεται μέσω της αντίστοιχης αναφοράς παραμέτρου που δέχεται.

Α. Κώδικας

```
public static void EncryptPassword(string Password, ref string EncryptedPW) {
        EncryptedPW = "";
        foreach (char c in Password)
        {
           int result = c + 5;
           if (result > 127) {
              result -= 128;
           }

           EncryptedPW += ((char)result).ToString();
        }
     }
```

Β. Πίνακας – περιπτώσεις ελέγχου

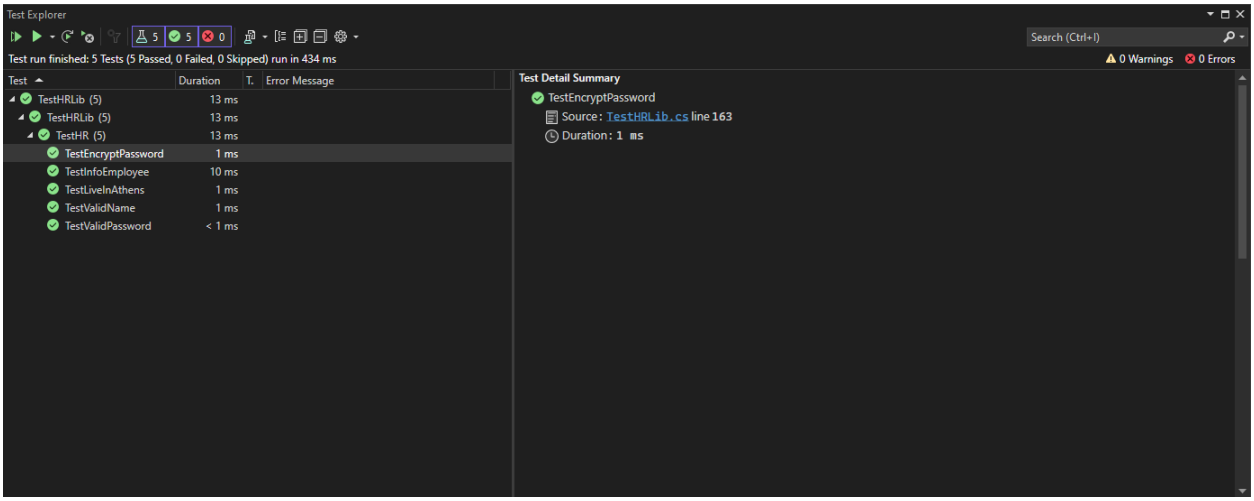
Περ.Ελ.	Δοκιμαστικά Δεδομένα Εισόδου	Αναμενόμενα Αποτελέσματα
#id	password	encrypted password
1	password	ufxx|twi
2	a3scr	f8xhw

Γ. Unit Tests

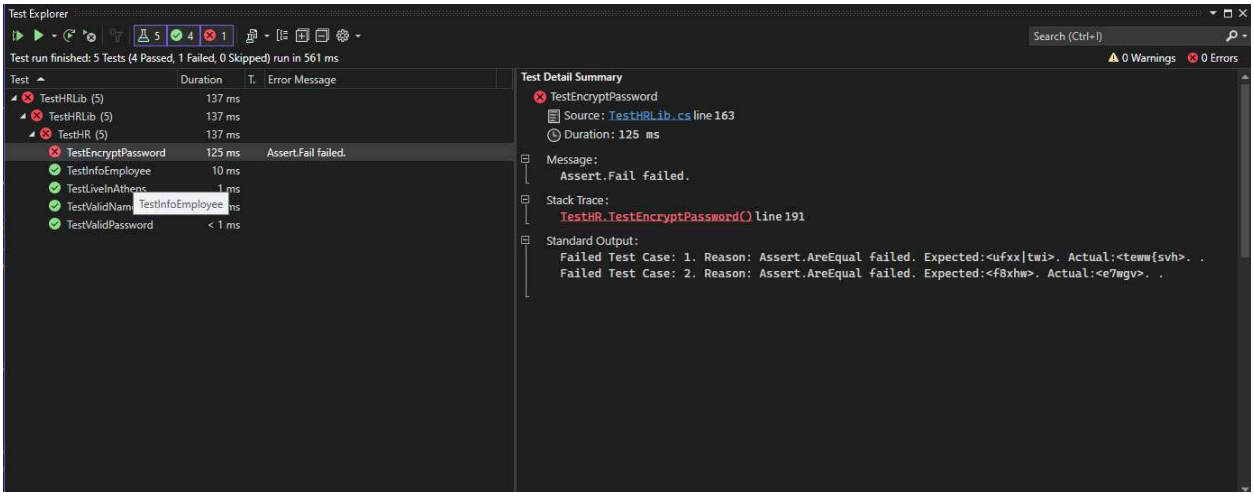
```
public void TestEncryptPassword()
     {
        object[,] testCases = {
           {1,"password", "ufxx|twi"},
           {2,"a3scr", "f8xhw"},
        };

        bool failed = false;

        for (int i = 0; i < testCases.GetLength(0); i++)
        {
           string encryptedPassword = "";
           HRLib.Hr.EncryptPassword((String)testCases[i, 1], ref encryptedPassword);

           try
           {
              Assert.AreEqual((string)testCases[i, 2], encryptedPassword);
           }
           catch (Exception e)
           {
              //Απέτυχε η περίπτωση
              failed = true;
              //Καταγράφουμε την περίπτωση ελέγχου που
              Console.WriteLine("Failed Test Case: {0}. Reason: {1}.", (int)testCases[i, 0], e.Message);
           };
        }

        //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
        if (failed) Assert.Fail();
     }
```

Δ. Αναφορές Ελέγχου

Με σωστά δεδομένα:



Αναφορές σφαλμάτων:



4. CheckPhone(string Phone, ref int TypePhone, ref string InfoPhone)

Η μέθοδος αυτή δέχεται ως κείμενο έναν αριθμό και ελέγχει αν αντιστοιχεί σε τηλέφωνο. Τα αποτελέσματα επιστρέφονται μέσω των αντίστοιχων αναφορών παραμέτρου που δέχονται. Συγκεκριμένα, αν είναι σταθερό τηλέφωνο, επιστρέφει ως τύπο τηλεφώνου (TypePhone) μηδέν (0) και ως πληροφορίες τηλεφώνου (InfoPhone) τη ζώνη που ανήκει ο αριθμός τηλεφώνου (8 ζώνες). Αν είναι κινητό τηλέφωνο, επιστρέφει ως τύπο τον αριθμό ένα (1) και ως πληροφορίες την εταιρεία κινητής τηλεφωνίας. Διαφορετικά, αν δεν είναι έγκυρος αριθμός τηλεφώνου, επιστρέφει ως τύπο τον αριθμό -1 και null στις πληροφορίες.

Α. Κώδικας

```
public static void CheckPhone(string phone, ref int phoneType, ref string phoneInfo)
    {

        if (phone.Length != 10 || !phone.All(char.IsDigit))
        {
            // Για να είναι τηλέφωνο πρέπει να έχει 10 ψηφία
            phoneType = -1;
            phoneInfo = null;
        }
        else  if (IsLandline(phone))
        {
            // Έλεγχος σταθερού τηλεφώνου
            phoneType = 0;
            phoneInfo = GetLandlineZone(phone);
        }
        else if (IsMobile(phone))
        {
            // Έλεγχος κινητού τηλεφώνου
            phoneType = 1;
            phoneInfo = GetMobileCompany(phone);
        }
        else
        {
            phoneType = -1;
            phoneInfo = null;
        }
    }

    public static bool IsLandline(string phone)
    {
        // Θα πρέπει το πρώτο ψηφίο να είναι 2 και το επόμενο από 1 έως και 8
        return phone[0] == '2' && phone[1] >= '1' && phone[1] <= '8';
    }

    public static string GetLandlineZone(string phone)
    {
        // Οι ζώνες χωρίζονται ανάλογα με το 2ο ψηφίο
        char secondChar = phone[1];
        switch (secondChar)
        {
            case '1':
                return "Μητροπολιτική Περιοχή Αθήνας – Πειραιά";

            case '2':
                return "Ανατολική Στερεά Ελλάδα, Αττική, Νησιά Αιγαίου";

            case '3':
                return "Κεντρική Μακεδονία";

            case '4':
                return "Θεσσαλία, Δυτική Μακεδονία";

            case '5':
```

```
                return "Θράκη, Ανατολική Μακεδονία";

            case '6':
                return "Ήπειρος, Δυτική Στερεά Ελλάδα, Δυτική Πελοπόννησος, Ιόνια Νησιά";

            case '7':
                return "Ανατολική Πελοπόννησος, Κύθηρα";

            case '8':
                return "Κρήτη";

            default:
                return "Άγνωστο";
        }
    }

    public static bool IsMobile(string phone)
    {
        return phone.StartsWith("69") && (phone[2] >= '0' && phone[2] <= '9'));
    }

    public static string GetMobileCompany(string phone)
    {
        // Επιστρέφει τον πάροχο του κινητού
        char thirdChar = phone[2];
        if (thirdChar == '0' || thirdChar == '3' || thirdChar == '9')
        {
            return "Nova";
        }
        else if (thirdChar == '4' || thirdChar == '5')
        {
            return "Vodafone";
        }
        else if (thirdChar == '7' || thirdChar == '8')
        {
            return "Cosmote";
        }
        else
        {
            return "Unknown";
        }
    }
```

B. Unit Tests

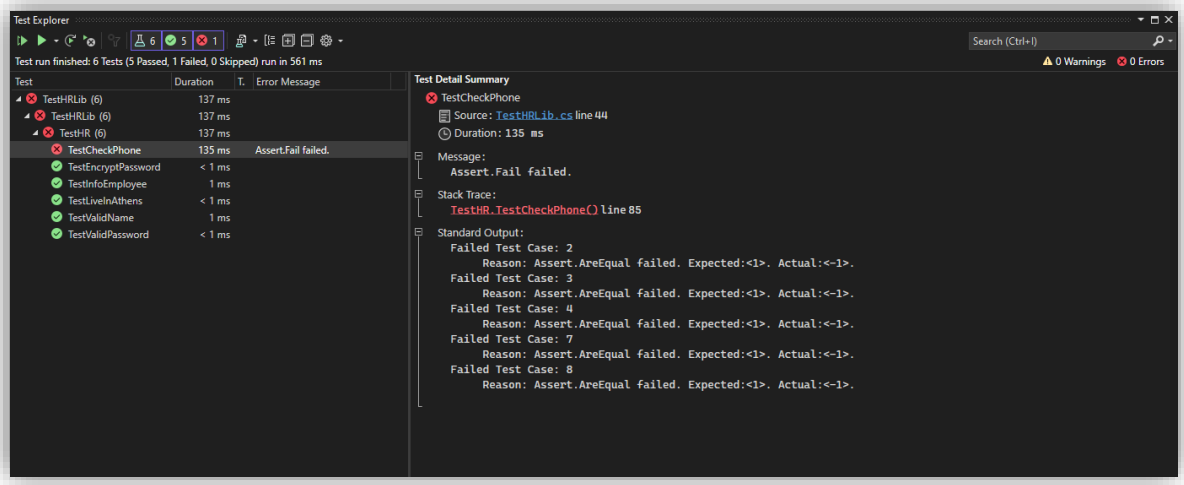
Περ.Ελ.	Δοκιμαστικά Δεδομένα Εισόδου	Αναμενόμενα Αποτελέσματα	
#id	phone	TypePhone	infoPhone
1	2101234567	0	Μητροπολιτική Περιοχή Αθήνας – Πειραιά
2	6940123456	1	Vodafone
3	6950123456	1	Vodafone
4	6970123456	1	Cosmote
5	23456	-1	null
6	2a34567890	-1	null
7	6923123456	1	Unknown
8	6900123456	1	Nova

Γ. Unit Tests

```
public void TestCheckPhone()
    // CheckPhone(string Phone, ref int TypePhone, ref string InfoPhone)
```

```
{
    object[,] testcases =
    {
        {1, "2101234567", 0,  "Μητροπολιτική Περιοχή Αθήνας – Πειραιά"},
        {2, "6940123456", 1, "Vodafone"},
        {3, "6950123456", 1, "Vodafone"},
        {4, "6970123456", 1, "Cosmote"},
        {5, "23456", -1, null},
        {6, "2a34567890", -1, null},
        {7, "6923123456", 1, "Unknown"},
        {8, "6900123456", 1, "Nova"}
    };


    bool failed = false;

    for (int i = 0; i < testcases.GetLength(0); i++)
    {
        try
        {
            // Μεταβλητές μεθόδου ref για να αποθηκευτούν οι τιμές που θα επιστρέψει
            int validPhoneType = -1;
            string phoneInfo = "";

            // Δίνουμε τα στοιχεία του test case
            HRLib.Hr.CheckPhone((string)testcases[i, 1], ref validPhoneType, ref phoneInfo);

            // Δίνουμε τις παραμέτρους που θέλουμε να συγκριθούν ως τιμές
            Assert.AreEqual((int)testcases[i, 2], validPhoneType);
            Assert.AreEqual((string)testcases[i, 3], phoneInfo);

        }
        catch (Exception exception)
        {
            failed = true;
            Console.WriteLine("Failed Test Case: {0}\n \t Reason: {1} ", (int)testcases[i, 0], exception.Message);
        }
    }

    if (failed) Assert.Fail();
}
```

Δ. Αναφορές Ελέγχου



5. InfoEmployee(Employee Empl, ref int Age, ref int YearsOfExperience)

Η μέθοδος αυτή δέχεται ως παραμέτρους τα στοιχεία ενός υπαλλήλου, και επιστρέφει την ηλικία του και τα χρόνια προϋπηρεσίας στην εταιρεία. Τα αποτελέσματα επιστρέφονται μέσω των αντίστοιχων αναφορών παραμέτρου που δέχονται.

Α. Κώδικας

```
public static void InfoEmployee(Employee Empl, ref int Age, ref int YearsOfExperience)
      {
         Age = (DateTime.Now - Empl.Birthday).Days / 365;
         YearsOfExperience = (DateTime.Now - Empl.HiringDate).Days / 365;
      }
```

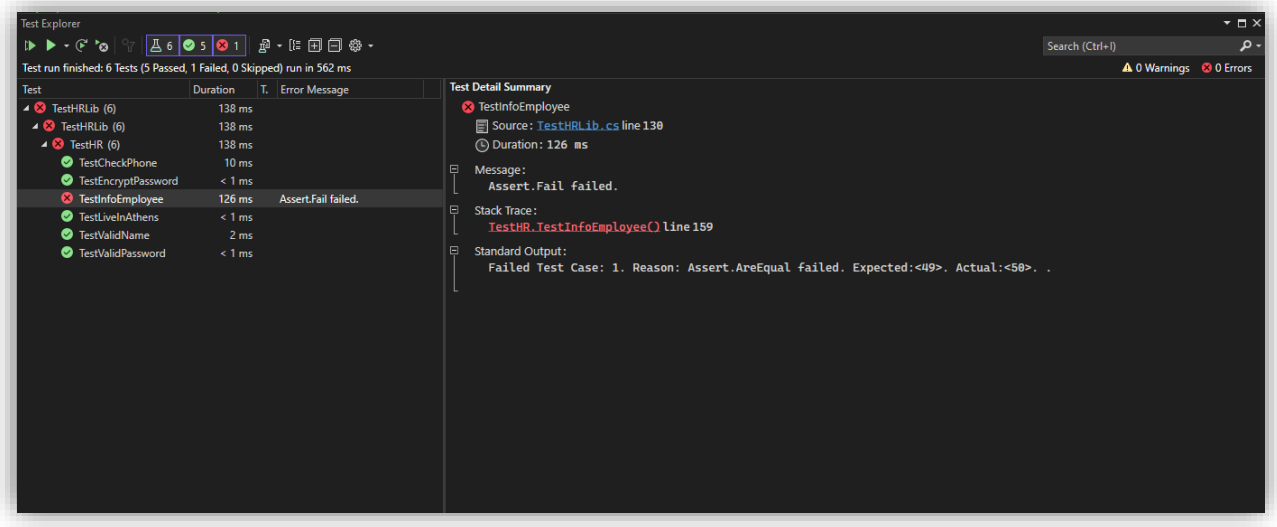
Β. Πίνακας – περιπτώσεις ελέγχου

Περ.Ελ.	Δοκιμαστικά Δεδομένα Εισόδου		Αναμενόμενα Αποτελέσματα	
#id	Employee EmplX		Age	YearsOfExperience
1	object[,] testCases = {{1, new Employee("Dimitrios Giannakopoulos", "1", "222", new DateTime(1974, 6, 22), new DateTime(2012, 12, 12)), 49, 11}};		49	11

Γ. Unit Tests

```
public void TestInfoEmployee()
      {
         object[,] testCases = {
            {1, new Employee("Dimitrios Giannakopoulos", "1", "222", new DateTime(1974, 6, 22), new DateTime(2012, 12, 12)), 49,
11}
         };

         bool failed = false;

         for (int i = 0; i < testCases.GetLength(0); i++)
         {
            int age = 0;
            int yearsOfExp = 0;
            HRLib.Hr.InfoEmployee((Employee)testCases[i, 1], ref age, ref yearsOfExp);

            try
            {
               Assert.AreEqual((int)testCases[i, 2], age);
               Assert.AreEqual((int)testCases[i, 3], yearsOfExp);
            }
            catch (Exception e)
            {
               //Απέτυχε η περίπτωση
               failed = true;
               //Καταγράφουμε την περίπτωση ελέγχου που απέτυχε
               Console.WriteLine("Failed Test Case: {0}. Reason: {1}.", (int)testCases[i, 0], e.Message);
            };
         }

         //Στην περίπτωση που κάποια περίπτωση ελέγχου απέτυχε, πέταξε εξαίρεση.
         if (failed) Assert.Fail();
      }
```

Δ. Αναφορές Ελέγχου



6. int LiveInAthens(Employee[] Empls)

Η συνάρτηση αυτή επιστρέφει το πλήθος των υπαλλήλων που κατοικούν στην Αθήνα. Η συγκεκριμένη συνάρτηση κάνει κλήση της συνάρτησης CheckPhone.

Α. Κώδικας

```
public static int LiveInAthens(Employee[] Empls)
    {
        int count = 0;

        foreach (Employee Empl in Empls)
        {
            int PhoneType = -1;
            string phoneInfo = "";
            CheckPhone(Empl.HomePhone, ref PhoneType, ref phoneInfo);

            if (PhoneType == 0 && phoneInfo.Contains("Μητροπολιτική Περιοχή Αθήνας – Πειραιά"))
            {
                // Ελέγχουμε εάν ο υπάλληλος ζει στην Αθήνα
                count++;
            }
        }
        return count;
    }
```

Β. Πίνακας – περιπτώσεις ελέγχου

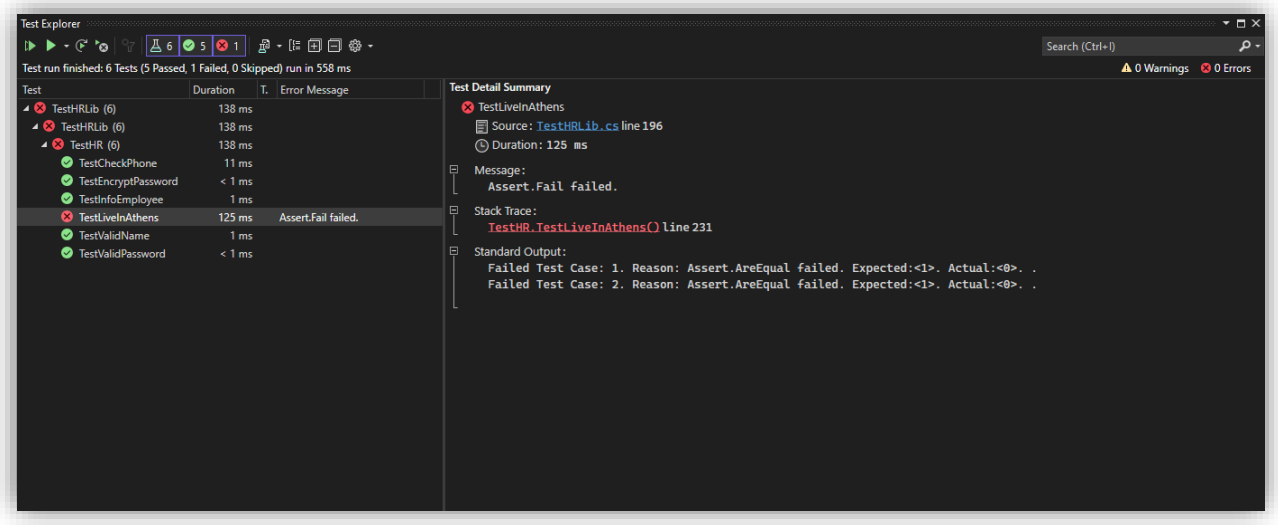
Περ Ελ	Δοκιμαστικά Δεδομένα Εισόδου	Αναμενόμενα Αποτελέσματα
#id	Employeep[] Empls	πλήθος
1	Employee[] arrayEmployee1 = {new Employee("Maria Papadopoulou", "2105551234", "6978123456", new DateTime(1985, 3, 15), new DateTime(2010, 7, 1)), new Employee("Alex Kwnstantinou", "2310333222", "6942001000", new DateTime(1990, 12, 5), new DateTime(2018, 11, 3))};	1
2	Employee[] arrayEmployee2 = { new Employee("Nikos Stathopoulos", "2106789456", "6934567890", new DateTime(1978, 9, 22), new DateTime(2015, 2, 10)) };	0

Γ. Unit Tests

```
public void TestLiveInAthens()
    {
        Employee[] arrayEmployee1 = {
            new Employee("Maria Papadopoulou", "2105551234", "6978123456", new DateTime(1985, 3, 15), new DateTime(2010, 7,
1)),
            new Employee("Alex Kwnstantinou", "2310333222", "6942001000", new DateTime(1990, 12, 5), new DateTime(2018, 11, 3))
        };

        Employee[] arrayEmployee2 = {
            new Employee("Nikos Stathopoulos", "2106789456", "6934567890", new DateTime(1978, 9, 22), new DateTime(2015, 2,
10))
        };


        object[,] testcases = {
            {1, arrayEmployee1, 1},
            {2, arrayEmployee2, 1}
        };

        bool failed = false;

        for (int i = 0; i < testcases.GetLength(0); i++)
        {
            int res = HRLib.Hr.LiveInAthens((Employee[])testcases[i, 1]);

            try
            {
                Assert.AreEqual((int)testcases[i, 2], res);
            }
            catch (Exception ex)
```

```
            {
                failed = true;
                Console.WriteLine("Failed Test Case: {0}. Reason: {1}.", (int)testcases[i, 0], ex.Message);

            }
        }

        if (failed) Assert.Fail();
    }
```

Δ. Αναφορές Ελέγχου



ΔΟΜΗ ΔΕΔΟΜΕΝΩΝ

Employee με πεδία:

- string Name (όνομα του υπαλλήλου)
- string HomePhone (σταθερό τηλέφωνο)
- string MobilePhone (κινητό τηλέφωνο)
- datetime Birthday (ημερομηνία γέννησης)
- datetime HiringDate (ημερομηνία πρόσληψης)

Α. Κώδικας

```
public struct Employee
   {
      public string Name;
      public string HomePhone;
      public string MobilePhone;
      public DateTime Birthday;
      public DateTime HiringDate;

      public Employee(string name, string homePhone, string mobilePhone, DateTime birthday, DateTime hiringDate)
      {
         Name = name;
         HomePhone = homePhone;
         MobilePhone = mobilePhone;
         Birthday = birthday;
         HiringDate = hiringDate;
      }
   }
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η βιβλιοθήκη HRLib αναπτύχθηκε με σκοπό την αποτελεσματική διαχείριση του τμήματος προσωπικού μιας εταιρείας, παρέχοντας λειτουργίες ελέγχου για ονόματα, κωδικούς πρόσβασης, τηλέφωνα και πληροφορίες υπαλλήλων. Οι περιπτώσεις ελέγχου και οι ελέγχοι μονάδων διασφαλίζουν τη σωστή λειτουργία και τη σταθερότητα των μεθόδων, ενώ οι αναφορές ελέγχου καταγράφουν λεπτομερώς τυχόν σφάλματα. Τα στιγμιότυπα οθόνης από το Test Explorer οπτικοποιούν τα αποτελέσματα, συμβάλλοντας στην αποτελεσματική διόρθωση πιθανών σφαλμάτων.