Εισαγωγή Στα Ενσωματωμένα Συστήματα

Καραπιπεράκης Εμμανουήλ

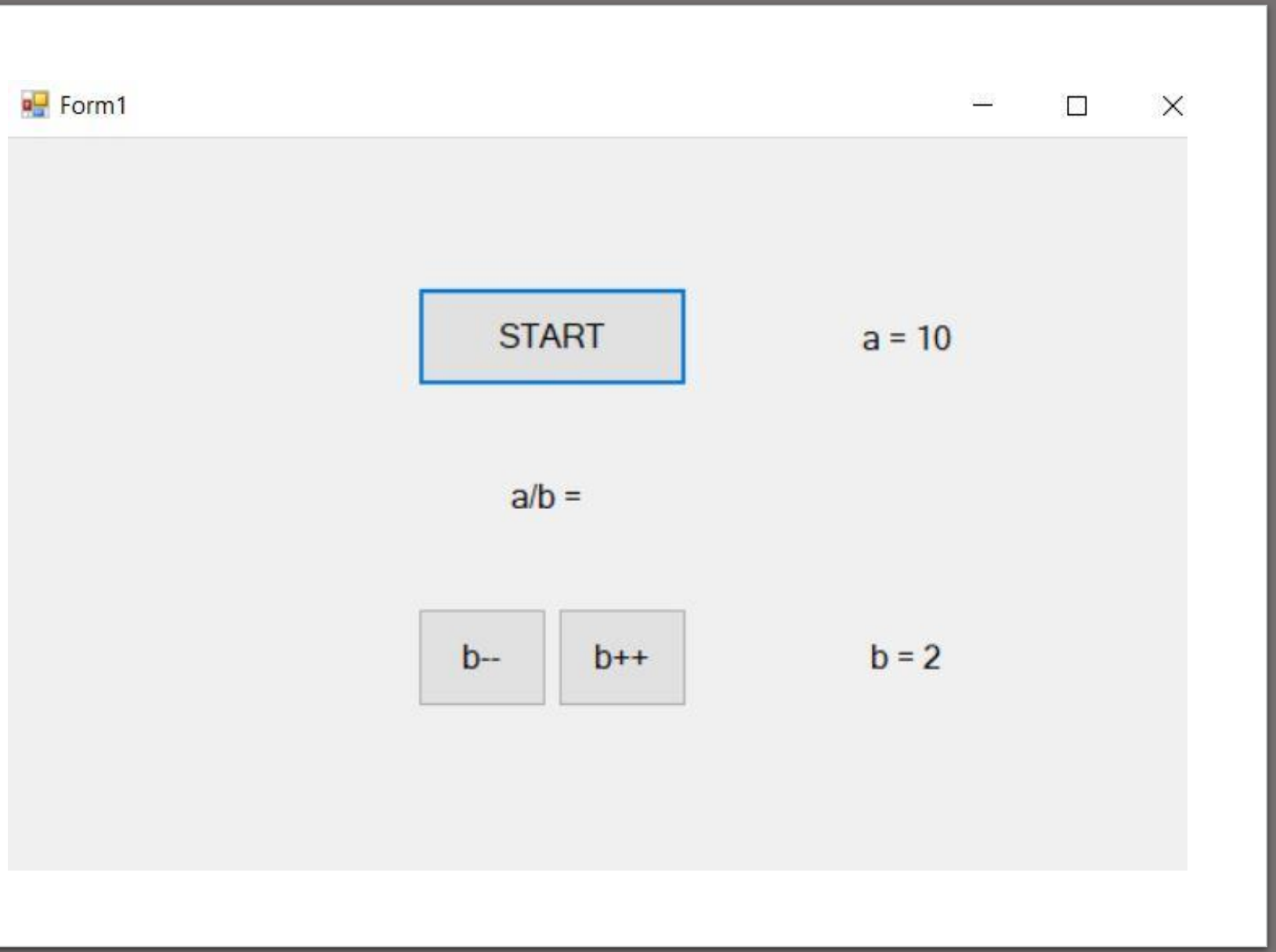
A.M:2022201800075

ΠΕΡΙΕΧΟΜΕΝΑ

Handling Exceptions from async Task Methods

Η διαχείριση των εξαιρέσεων αποτελεί σημαντικό κομμάτι κάθε σχεδίασης. Εκτός από την περίπτωση επιτυχίας θα πρέπει να γίνονται οι κατάλληλες ενέργειες, με σκοπό την ορθή αντιμετώπιση των περιπτώσεων που απέτυχαν.

Ας πάρουμε για παράδειγμα την ακέραια διαίρεση 2 αριθμών a / b. Για b <> 0 δεν υπάρχει κάποιο πρόβλημα και ο υπολογισμός της πράξης γίνεται κανονικά. Για b = 0 όμως θα πρέπει να υπάρχει ένα exception που να χειρίζεται αυτή την περίπτωση καθώς είναι ανεπιθύμητη. Έτσι δημιουργούμε ένα exception το οποίο στην περίπτωση όπου b == 0 δεν εκτελεί την πράξη αλλά επιστρέφει μήνυμα σφάλματος.



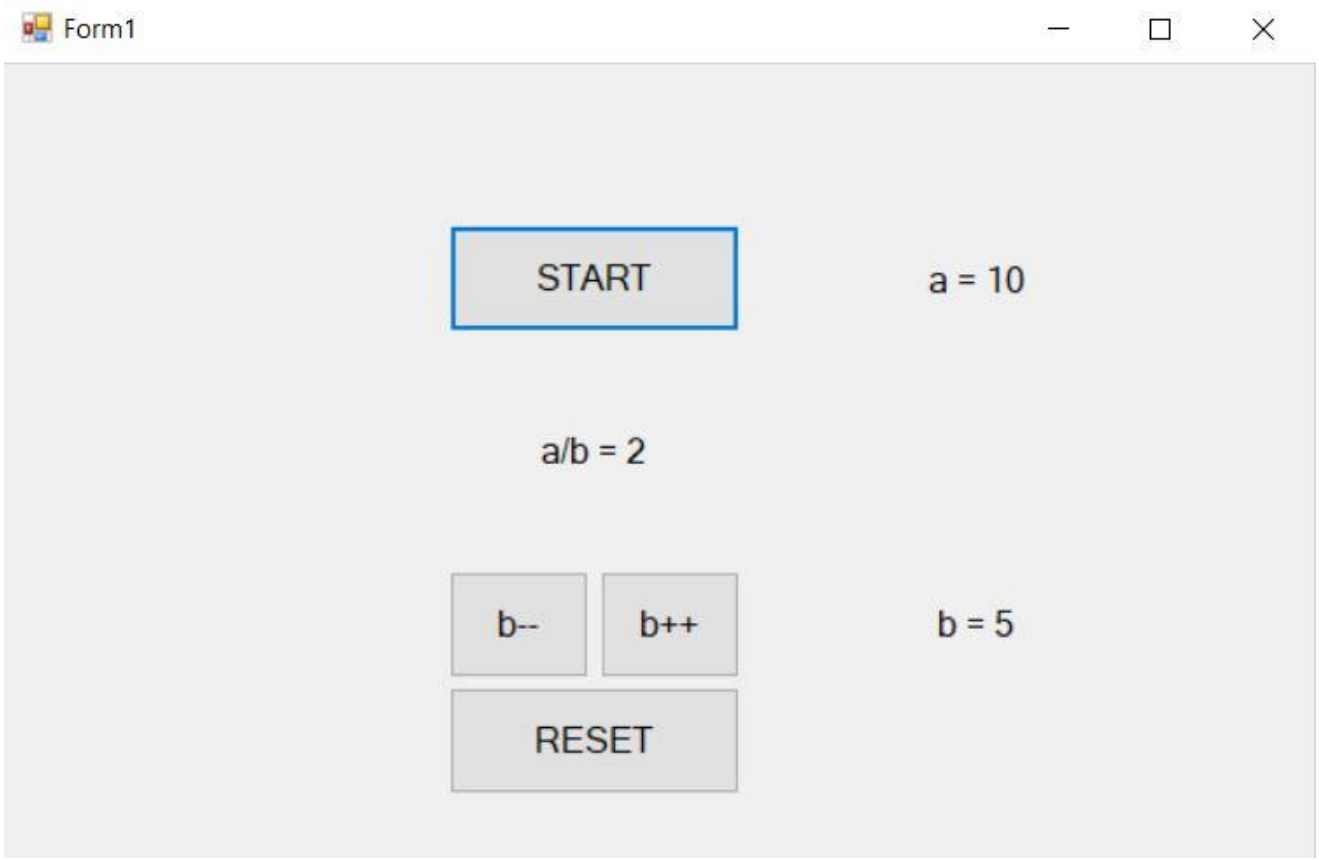
Windows Forms App

- Όπως φαίνεται και στην προηγούμενη εικόνα το πρόγραμμα αρχικά αποτελείται από 3 κουμπιά τα οποία ουσιαστικά αντιστοιχούν σε 3 ασύγχρονες μεθόδους
  - **START method:** Υπολογίζει το αποτέλεσμα της πράξης a/b
  - **b--method:** Μειώνει την τιμή της μεταβλητής b κατά 1
  - **b++method:** Αυξάνει την τιμή της μεταβλητής b κατά 1

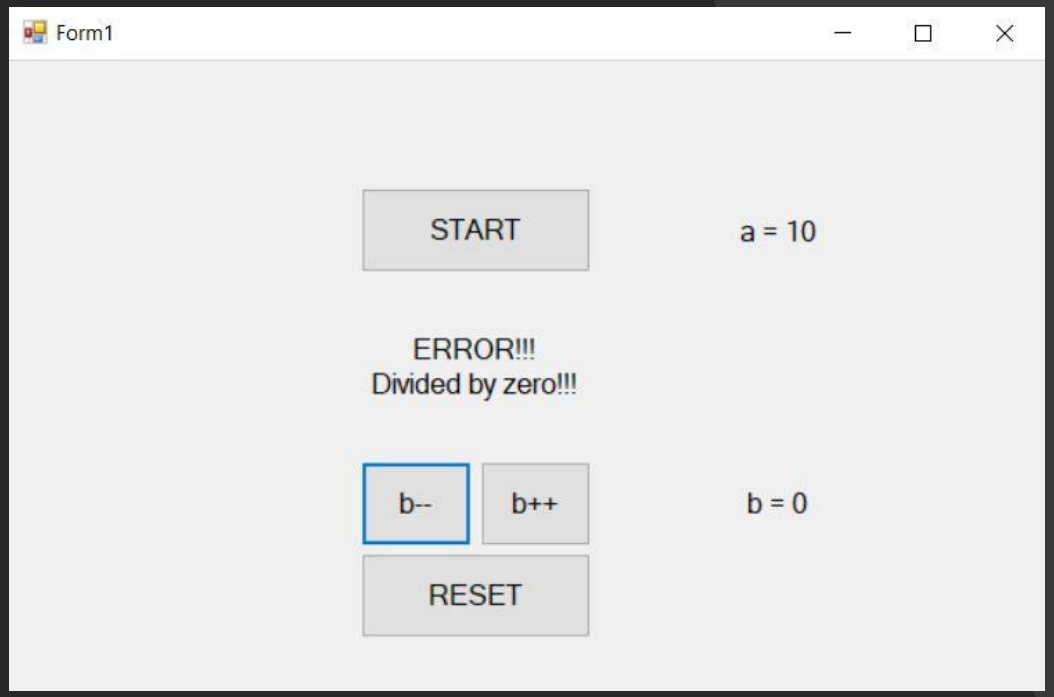
- By default η μεταβλητή a είναι αρχικοποιημένη στην τιμή 10 και η μεταβλητή b στην τιμή 2

- Στην μέθοδο START έχουμε προσθέσει ένα delay 1.5 δευτερόλεπτο χρησιμοποιώντας τη συνάρτηση BigTask και στις μεθόδους b-- και b++ έχουμε προσθέσει ένα μικρότερο delay της τάξεως των 0.2 δευτερολέπτων. Επειδή οι συναρτήσεις είναι ασύγχρονες αυτό σημαίνει πως καθώς εκτελείται μια μέθοδος μπορούμε παράλληλα να εκτελέσουμε κάποια άλλη. Για παράδειγμα αν πατήσουμε το κουμπί START στο διάστημα μέχρι να υπολογιστεί το αποτέλεσμα της πράξης a/b μπορούμε να αυξομειώσουμε την τιμή της μεταβλητής b πατώντας τα αντίστοιχα κουμπιά.

- Όπως φαίνεται για είσοδο διαφορετική από '0' ο υπολογισμός της πράξης γίνεται κανονικά και το αποτέλεσμα φαίνεται κάτω από το κουμπί "START". Το κουμπί "Reset" υπάρχει για να επαναφέρουμε το b στην προκαθορισμένη του τιμή, δηλαδή την τιμή 2.
- Τώρα στην περίπτωση όπου b = 1, αν πατήσουμε το κουμπί "START" και στη συνέχεια το κουμπί "b--" πριν ολοκληρωθεί η διεργασία του υπολογισμού της ακέραιας διαίρεσης θα δούμε πως το 'b' θα πάρει την τιμή '0'. Επομένως η πράξη δεν μπορεί να γίνει και θα εκτυπωθεί μήνυμα σφάλματος όπως φαίνεται στην επόμενη εικόνα



- Για την αντιμετώπιση της περίπτωσης όπου ο παρονομαστής είναι 0 κατασκευάσαμε τη μέθοδο ***public int Division(int a,int b)***
- Η παραπάνω μέθοδος καλείται μέσα από το σώμα της συνάρτησης περνώντας ως όρισμα τις τιμές των μεταβλητών a και b και επιστρέφει το αποτέλεσμα της ακέραιας διαίρεσης πάνω στην ασύγχρονη μέθοδο που περιγράφεται από το κουμπί "START".
- Με αυτό τον τρόπο διασφαλίζουμε πως αν η μεταβλητή 'b' μετά από ένα σήμα διακοπής λάβει την τιμή '0' θα μπορέσουμε να διαχειριστούμε επιτυχώς αυτή την περίπτωση.



Εξήγηση Κώδικα

Στις επόμενες εικόνες φαίνονται η συνάρτηση που αντιστοιχεί στο κουμπί "START" ή οποία καλεί ασύγχρονα τη συνάρτηση που εκτελεί την πράξη της διαίρεσης και αν υπάρχει exception(b=0) το κάνει catch, η συνάρτηση BigTask, δηλαδή η καθυστέρηση που προαναφέραμε και η συνάρτηση που υπολογίζει και επιστρέφει το αποτέλεσμα της ακέραιας διαίρεσης πάνω, στην ασύγχρονη μέθοδο που προαναφέραμε.

```
static void BigTask()
{
    Thread.Sleep(1500);
}
```

```
private async void button1_Click(object sender, EventArgs e)
{
    button4.Visible = false;
    await Task.Run(new Action(BigTask));

    try
    {
        int result =  await Task<int>.Run(() => { return Division(a,b); });
        label1.Text = "a/b = " + Convert.ToString(result);
        button4.Visible = true;
    }
    catch (DivideByZeroException)
    {
        label1.Text = "ERROR!!!\nDivided by zero!!!";
        button4.Visible = true;
    }
}

1 reference
public int Division(int a,int b)
{
    result = a / b;
    return result;
}
```

```
1 reference
private async void button3_Click(object sender, EventArgs e)
{

    await Task.Run(new Action(SmallTask));
    b++;
    label3.Text = "b = " + Convert.ToString(b);
}

1 reference
private async void button2_Click(object sender, EventArgs e)
{

    await Task.Run(new Action(SmallTask));
    b--;
    label3.Text = "b = " + Convert.ToString(b);
}
```

```
3 references
static void SmallTask()
{
    Thread.Sleep(200);
}
```

- Αντίστοιχα τώρα φαίνονται οι εικόνες των κουμπιών b--,b++ που υλοποιούν την αυξομείωση της τιμής που έχει η μεταβλητή 'b'. Επίσης φαίνεται η συνάρτηση SmallTask που ανιστοιχεί στην μικρότερη καθυστέρηση για την ολοκλήρωση των 2 παραπάνω μεθόδων:

Βιβλιογραφία

- Concurrency in C# Cookbook