SARRERA

Visual Studio 2022 C# .NET

Idazmahairako interfaze garapena

PAAG 2023-2024 Ikasturtea

Sarrera. Idazmahairako interfaze garapena

Edukiak:

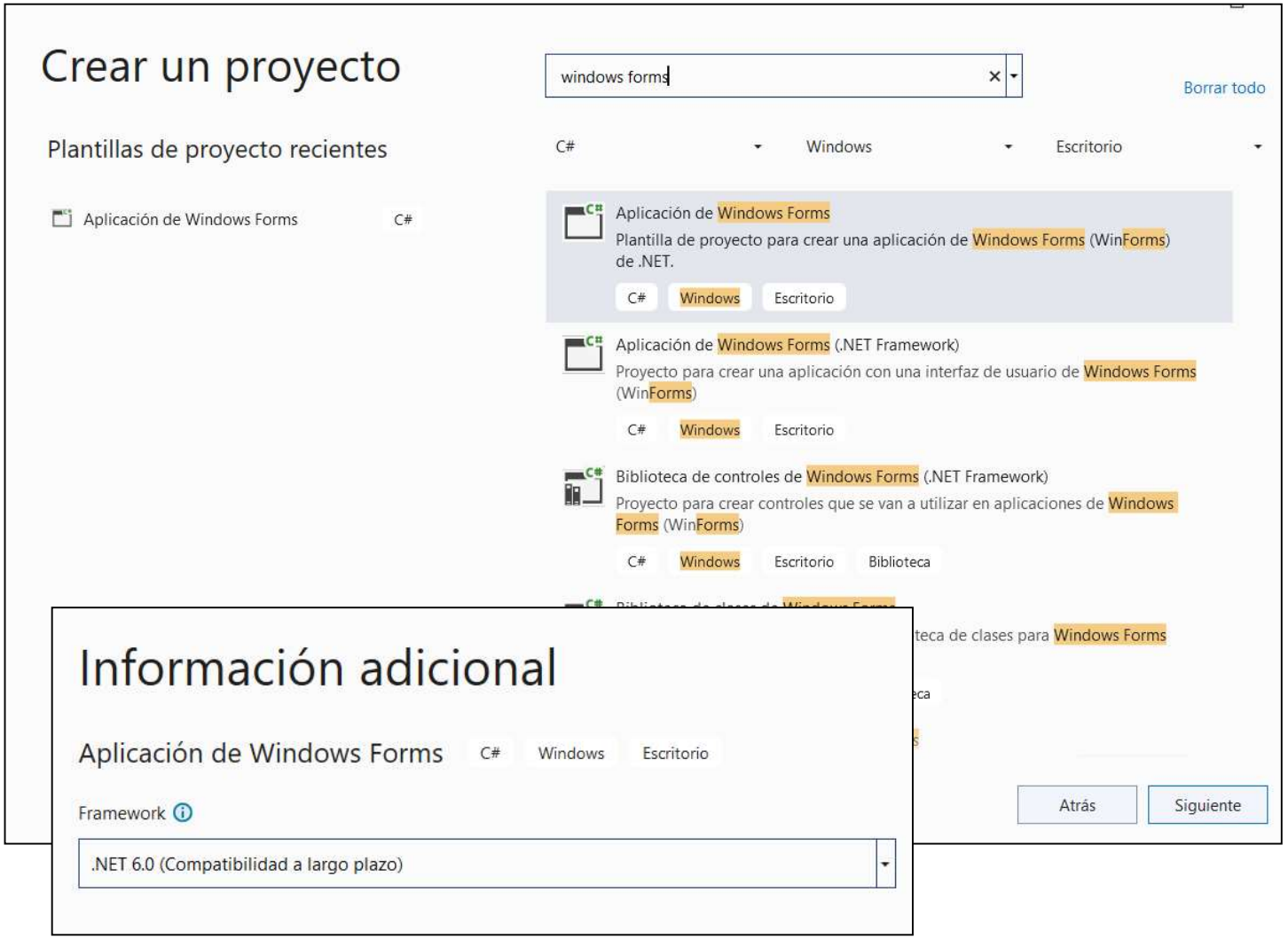
- Izenen eremuak, aldagaiak, motak
- Klaseak
  - Objektu bat sortu eta modelatu:
    - Bere propietateak (aldagaiak, get, set)
    - Bere portaera (metodoen bidez)
    - STATIC erabilera
    - Herentzia eta Polimorfismoa
    - Klase eta metodo birtualak
    - Interfazeak
- Baldintzazko egitura: IF (baldintza){…}ELSE{…}
- Erroreak kontrolatu EXCEPTIONS: TRY …CATCH ..FINALLY

Sarrera. Idazmahairako interfaze garapena

- Egitura errepikakorra: FOR (hasiera; amaiera; gehiketa){…}
  FOREACH (var1 IN var2)
  DO {…} WHILE (baldintza)
- Aldagaien ikuspena: PUBLIC, PRIVATE, PROTECTED
- Teklak kontrolatzen: zenbakiak bakarrik onartu, koma bakarra onartu, lehenengo karakterra koma bada, “0,” jarri
- Ebentoak
- Parametroak pasatzen
- ARRAY eta ZERRENDAK

Sarrera. Idazmahairako interfaze garapena

Instalatzeko (Visual Studio 2022 Community):
https://visualstudio.microsoft.com/es/vs/

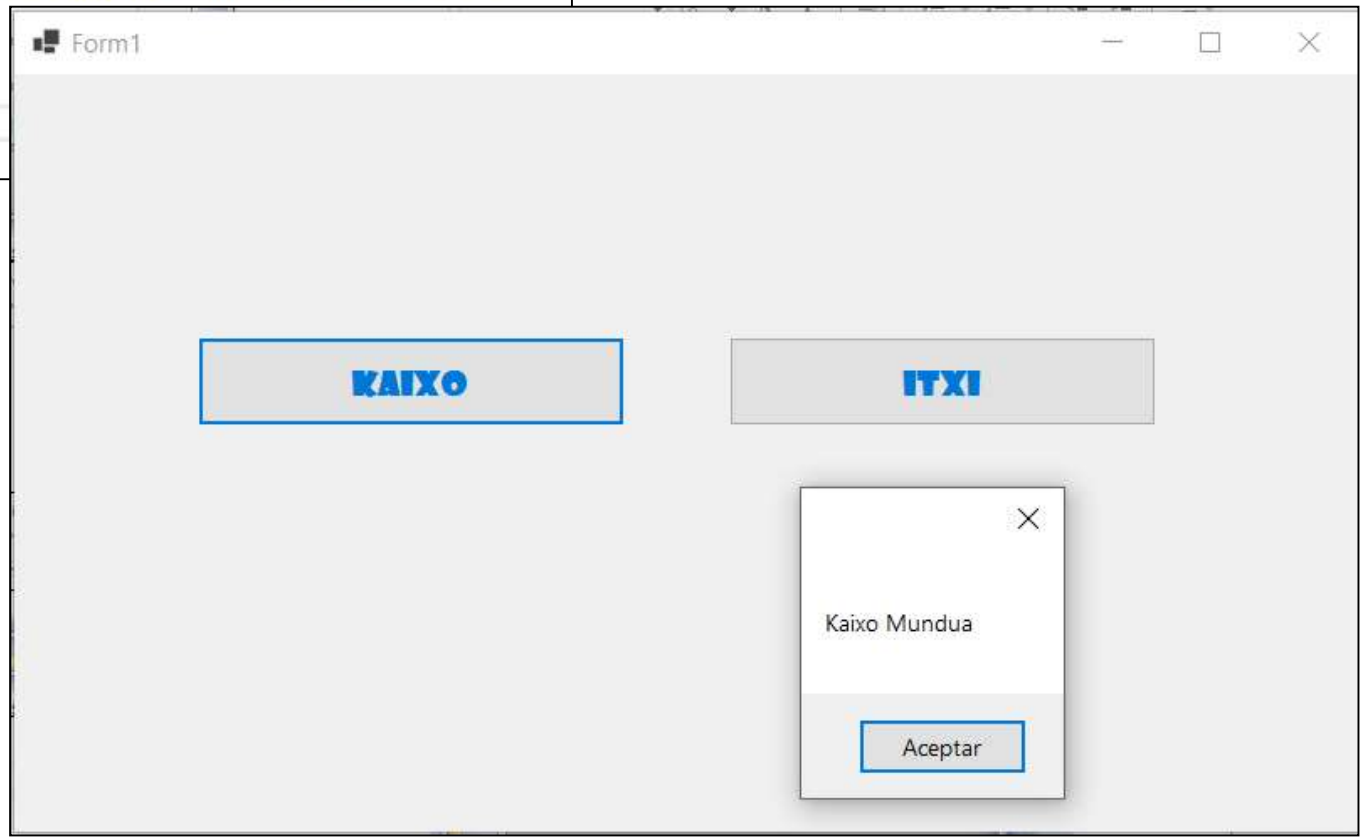


Sarrera. Idazmahairako interfaze garapena

Adibidea: Kaixo Mundua (Windows Forms motako aplikazioa)

```
1 referencia
private void btnKaixo_Click(object sender, EventArgs e)
{
    MessageBox.Show("Kaixo Mundua");
}

1 referencia
private void btnItxi_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}
}
```



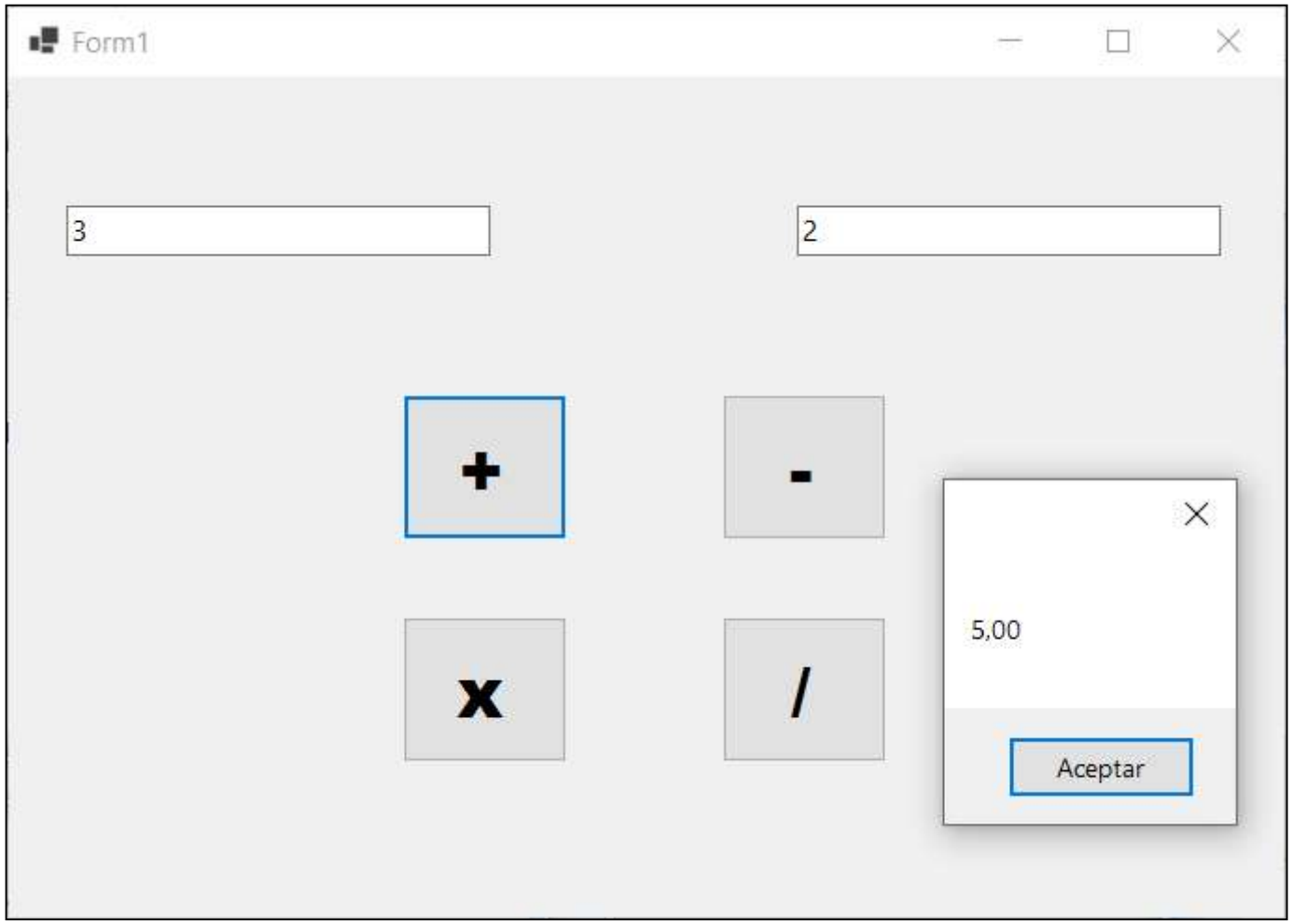
Tresnak ikusten ez diranean («cuadro de herramientas» leihoan ez dira agertzen)

->

eskuineko botoian klik egin – «elegir elementos» - «Restablecer»

Sarrera. Idazmahairako interfaze garapena

Adibidea: Oinarrizko kalkulagailua



Sarrera. Idazmahairako interfaze garapena

Objektuetara zuzendutako programazioa erabiliz: klasea definitzen da

```
namespace WinForms_Kalkulagailua
{
    internal class Kalkulagailua
    {
        //propietateak
        public float Zenbaki1 { get; set; }
        public float Zenbaki2 { get; set; }

        //eraikitzailea
        public Kalkulagailua(float zenbaki1, float zenbaki2)
        {
            this.Zenbaki1 = zenbaki1;
            this.Zenbaki2 = zenbaki2;
        }

       //metodoak
        public float Gehiketa()
            { return Zenbaki1+ Zenbaki2; }
    }
}
```

Sarrera. Idazmahairako interfaze garapena

Objektuetara zuzendutako programazioa erabiliz: botoiaren ebentoa programatzen da:

```
    private void btnGehiketa_Click(object sender, EventArgs e)
     {
        //Objektua sortzen

        Kalkulagailua kalkulagailua = new Kalkulagailua(float.Parse(txtZenbaki1.Text),
float.Parse(txtZenbaki2.Text));

        //Objektuaren metodoa erabiltzen

        MessageBox.Show(kalkulagailua.Gehiketa().ToString("0.00"));

    }
```

Sarrera. Idazmahairako interfaze garapena

Ariketa: Beste botoiak programatu eta erroreak kontrolatu:

-Try ……Catch …..Finally egitura erabili: testu kaxan ez dela ezer idazten edota ez dela zenbakia. Erroreak kontrolatu

Sarrera. Idazmahairako interfaze garapena

- Zuzenean ez utzi zenbakia ez den ezer idazten

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)

{   if (!Char.IsDigit(e.KeyChar)                  // ez bada zenbakia
      && e.KeyChar != Convert.ToChar(Keys.Back)   // ez bada atzera
      && e.KeyChar != Convert.ToChar(Keys.Delete)// ez bada ezabatu
       && e.KeyChar != Convert.ToChar(","))       // ez bada koma
      {
         e.Handled = true;                                  // ebentoa kontrolatu
         return;                                                  // bueltatu
      }
      else
          {
             if (e.KeyChar == Convert.ToChar(","))  // koma bada
             {
                if (txtZenbaki1.Text.IndexOf(",") >= 0)  // jada badago koma bat
                {
                   e.Handled = true;            //ebentoa kontrolatu
                   return;                                 //bueltatu
                }
                  else
                {
                   if (txtZenbaki1.Text.Length == 0)  // lehenengo  koma aurretik 0 bat jartzeko
                     {
                       e.Handled = true;
                       SendKeys.Send("0,");
                     }
                }
             }
          }
}
```

Sarrera. Idazmahairako interfaze garapena

- Eta ez da utzi behar beste inondik kopiatzen ere:

  Horretarako propietate bat erabili daiteke:

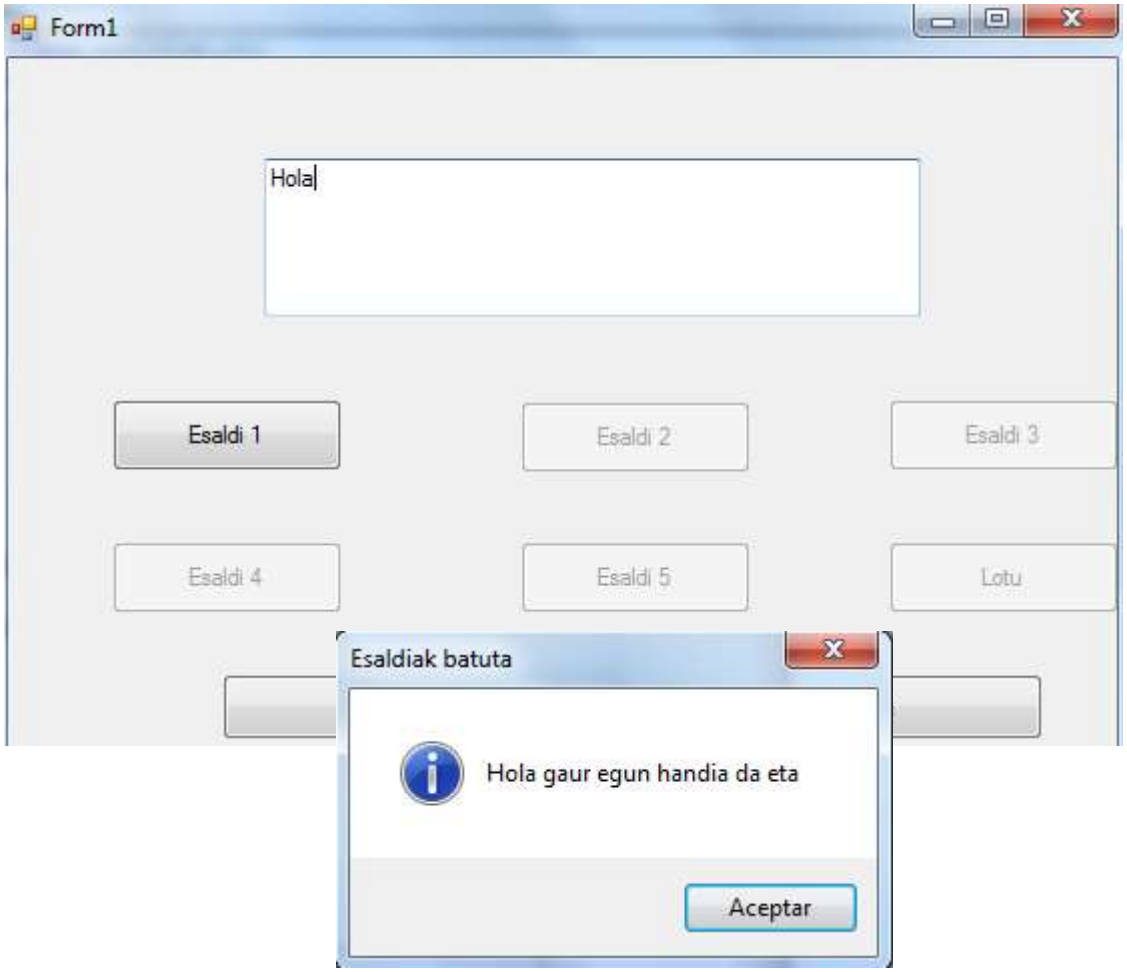
  - **ShortcutsEnabled** (Testu kaxa bakoitzean) **false** jarrita.

- Egin ditugun balidazio guztiak erabili behar dira, lehenengoa, testu kaxak hutsik ez direla ixten bermatzeko eta bigarrena, letrarik idazten ez uzteko

Sarrera. Idazmahairako interfaze garapena

Ariketak

Ariketa 1: Sortu esaldiak lotzen dituen aplikazioa. Hasieran “Esaldi1” botoia bakarrik aktibatuta dago. Testu kutxan lehenengo esaldia idatzi eta esaldi1 botoia sakatzen dugu, hau desaktibatua geratzen da eta bigarrena aktibatuta. Bostgarren botoia sakatu ondoren “lotu” botoia aktibatzen da. Hau zapaltzerakoan, esaldi guztiak agertuko dira, euren artean tarte zuri batekin. “Garbitu” zapaldu ondoren “Esaldi1” botoia aktibatu eta fokoa testu kutxara doa.



Klase bat egin, Esaldia: 2 propietate eta metodo bat eduki behar ditu

.

Propietate bat **EsaldiaBatuta:** irakurtzeko soilik, orduan get bakarrik dauka.

Sarrera. Idazmahairako interfaze garapena

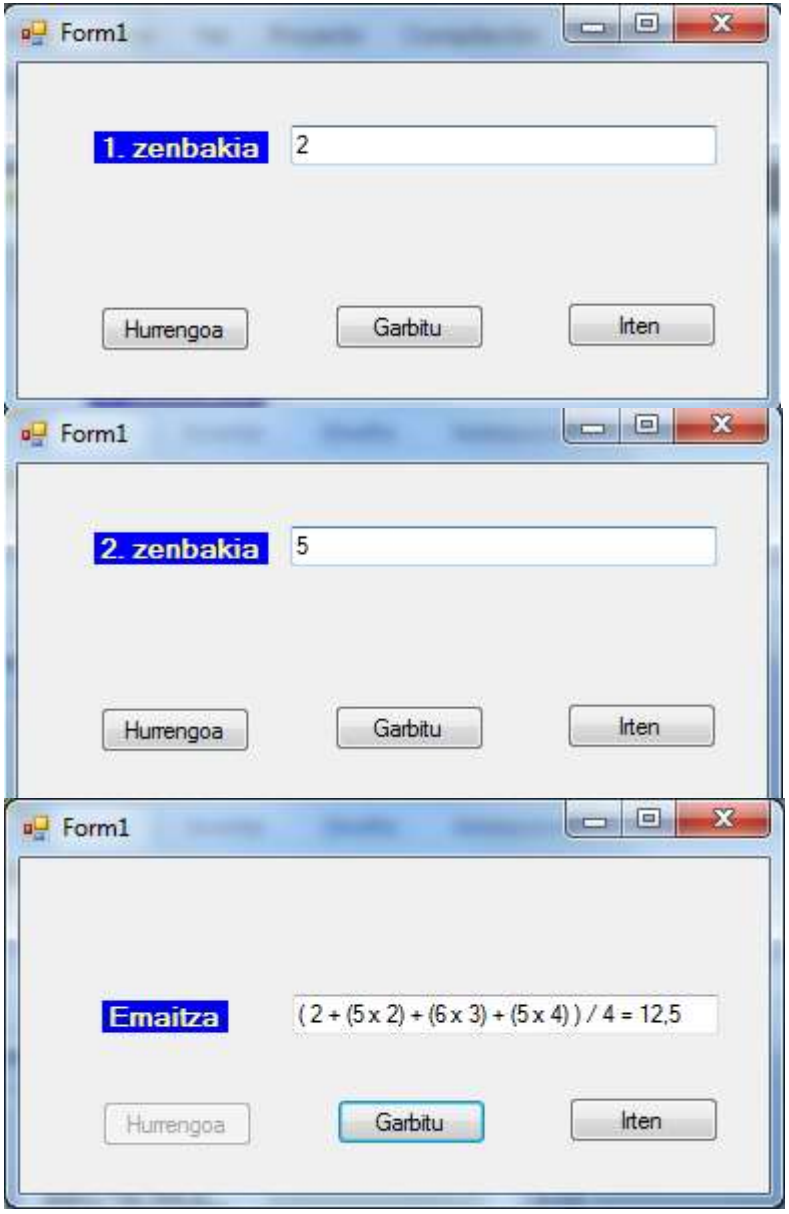
Ariketa 2: Hasieran 1. irudia ikusten da. Zenbaki bat sartu ondoren “Hurrengoa” botoia sakatzen dugu. “2.zenbakia” agertzen da etiketan eta beste zenbaki bat itxaroten geratzen da. Hiru eta laugarren zenbakiekin berdin egiten dugu. Azken zenbakia sartu eta “Hurrengoa” sakatuz gero 3. irudia agertuko da.

Formula: (a+2b+3c+4d)/4

egin den eragiketa agertzen da

“Garbitu” sakatuta hasierako modura itzultzen gara.

-Klase bat sortu zenbakiak gordetzeko eta eragiketa egiteko
-Switch case …. egitura erabili



Sarrera. Idazmahairako interfaze garapena

ARRAY

Array-ak balio asko gordetzeko erabiltzen dira. Dimentsio batekoak edo askokoak izan daitezke

```
char[] bokalak = new char[5] {'a', 'e', 'i', 'o', 'u'};
for (int i=0; i<5; i++)          // bokalak.Length erabili ahal da dimentsio bakarrarekin
        {
           Console.WriteLine("Elemento {0}: {1}", i, bokalak[i]);
           a= Console.ReadLine();
        }

int[,] matriz1 = new int[3, 2] {{1, 2}, {3, 8}, {4, 6}};     //3 zutabe, 2 lerro
for (int i=0; i<matriz1.GetLength(0); i++)                //GetLength(0) lehenengo dimentsioko luzera
        {
           Console.Write(“Zutabe {0}: ", i+1);
           for (int j=0; j<matriz1.GetLength(1); j++) //GetLength(1) bigarren dimentsioko luzera
           {
              Console.Write("{0} ", matriz1[i,j]);
           }
           Console.WriteLine();
           Console.WriteLine();
        }
         a=Console.ReadLine();
```

Sarrera. Idazmahairako interfaze garapena

Ariketa 2 - array: Ariketa aldatu array bat erabiliz:
Formularioan:

```
float[] zenbakiak = new float[4];
```

Klasean:
Propietatea array bat izango da:

```
 float[] zenbakiak = new float[4];
    public float[] Zenbakiak
        {
            get
            {
                return zenbakiak;
            }

            set
            {
                zenbakiak = value;
            }
        }
```

+ Metodoa formula aplikatzeko

LIST – ZERRENDA GENERIKOA

Zerrenda generikoa edozein motako objektuen zerrenda bat gordetzeko erabiltzen da:

Adibidea:
Klasea bat sortzen dugu:

```
Class Datuak
{
   public string Izena {get; set; }

   public Datuak(string izena)
    {this.Izena = izena;}
}
```

Programa nagusian, bi objektu gordetzen ditugu "zerrenda" batean

```
Static void Main()
{
  List<Datuak> datuak = new List<Datuak>();
          datuak.Add(new Datuak("Aulkia"));
          datuak.Add(new Datuak("Laranja"));
```

Consolan ikusten ditugu, "for" egitura erabiliz:

```
for (int i = 0; i < datuak.Count; i++)
        {
           Console.WriteLine(datuak[i].Izena);
        }
        Console.ReadLine();
```

Consolan ikusten ditugu, "foreach" egitura erabiliz:

```
 foreach (Datuak datua in datuak)
        {
           Console.WriteLine(datua.Izena);
        }
        Console.ReadLine();
```

Sarrera. Idazmahairako interfaze garapena

STATIC

Klase bat estatikoa izan daiteke(ezin da instantziatu). Klase ez estatikoetan propietate estatikoak edo metodo estatikoak topatu ditzakegu:

- Ez dira objetuen osagaiak eta beraz ez dira desberdinak objetu bakoitzentzat.
- Klasearen kideak dira eta beraz kopia bakarra existitzen da. Klasearen izena aurretik jarrita erabiltzen dira.

Adibidea:

```
Class Automobile
{ static int sizeOfGasTank;
public static int SizeOfGasTank
{
get { return sizeOfGasTank; }
}
public static void Drive()
 { }

// Other non-static fields and properties... }
```

Programa nagusian:

```
……..
Automobile.Drive();

int i = Automobile.SizeOfGasTank;

……
```

Sarrera. Idazmahairako interfaze garapena

Ariketa 2 - zerrenda: Ariketa aldatu “zerrenda”kin Formularioan Ariketa2 klaseko objektuak sortu eta gorde zerrenda batean eta azkenean emaitza eman.

List<Ariketa2> zenbakiak = New List<Ariketa2>();

Klaseak propietate bi ditu:

```
Class Ariketa2
{
public string Label { get; set; }
public double Zenbakia { get; set; }
}
```

eta metodo estatikoa

```
public static double eragiketa(List<ariketa2> objektuak)
        {
        }
```

Sarrera. Idazmahairako interfaze garapena

HERENTZIA eta POLIMORFISMOA

```
internal class A
   {
      public string Prop1 { get;set; }
      public virtual string Prop2 { get;set; }

      public virtual void Erakutsi()
      {
         Console.WriteLine("A klasea");
          Console.WriteLine("{0}, {1}", Prop1, Prop2);
      }
```

"Virtual" definitzen ditugu propietateak eta metodoak. Berengandik heredatzen duen klaseak **"override"** erabiliz metodo edo propietate birtual hauek inplementatzeko aukera du

HERENTZIA eta POLIMORFISMOA

```
internal class B : A
   {
      public string Prop3 { get; set; }
      public override string Prop2
      {
         get => base.Prop2;
         set
         {
            if (value.Length < 8)
            {
               Exception ex = new Exception("Ez da
zuzena. 8 karakter gutxienez.");
               throw ex;
            }
            else
            {
               base.Prop2 = value;
            }
         }
      }
      public override void Erakutsi()
      {
         Console.WriteLine("B klasea");
         Console.WriteLine("{0}, {1}, {2}", Prop1,
Prop2, Prop3);
      }
```

```
try {
    A objA = new A();
    objA.Prop1 = "pepe";
    objA.Prop2 = "20";
    objA.Erakutsi();
    B objB = new B();
    objB.Prop1 = "izaskun";
    objB.Prop2 =
"4733333333333333333333";
    objB.Prop3 = "Kortabitarte";
    objB.Erakutsi();
    Console.ReadLine();
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    Console.ReadLine();
}
```

Sarrera. Idazmahairako interfaze garapena

HERENTZIA eta POLIMORFISMOA

Ariketa 3: Sortu klase bat “Kontaktua”
propietate hauekin: Nan, Izena, Abizena, Emaila (“virtual”), izenOsoa (irakurtzeko soilik)
Eta metodo hau: Gorde (“virtual”) mezu bat ateratzen duena: “Kontaktua ondo gorde da”

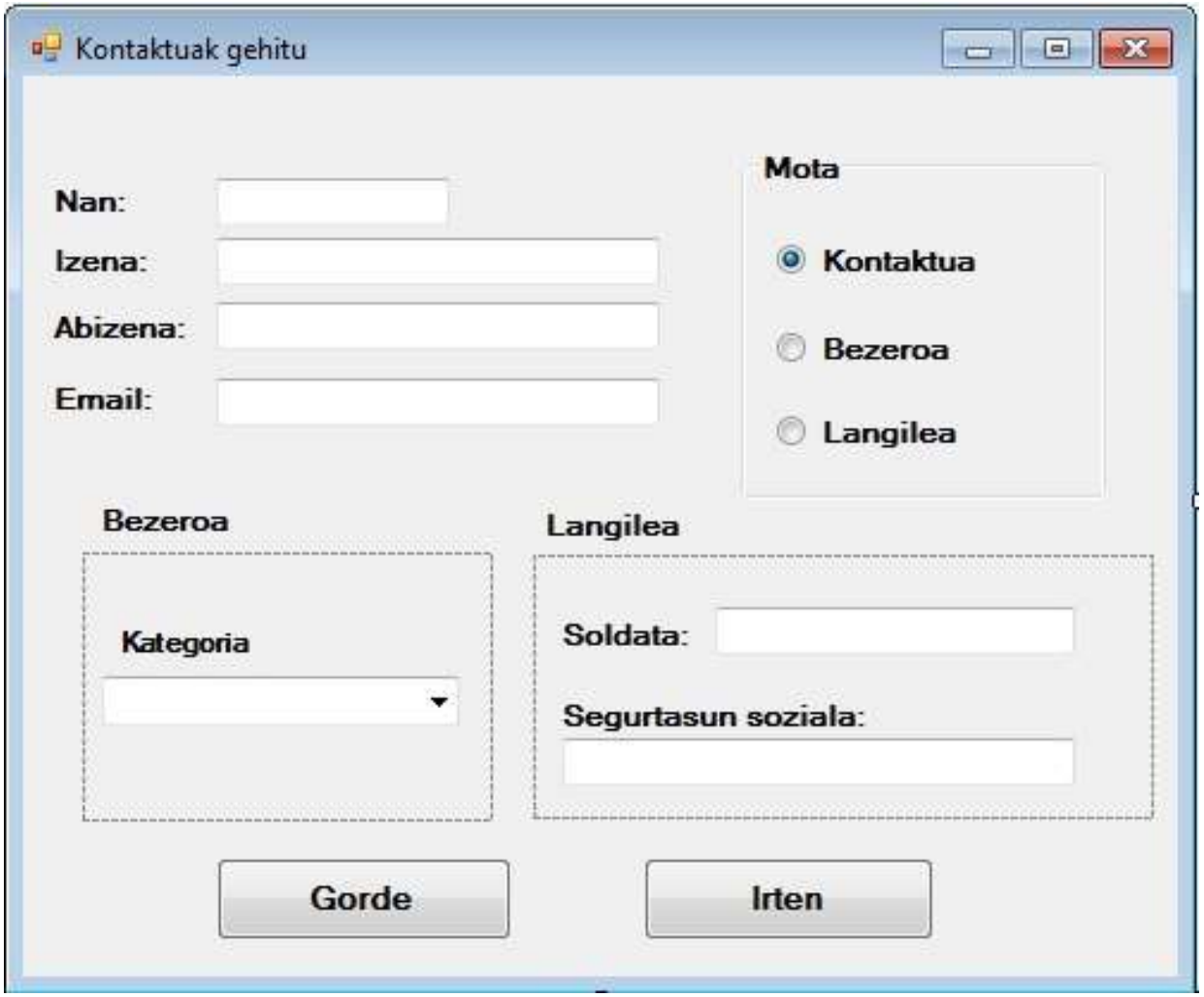
Beste klase bat sortu “Bezeroa”, aurreko klasetik heredatzen duena.
Gehi propietate bat: Kategoria
Eta metodo Gorde berriro definitzen du (override), mezua aldatuta “Bezeroa ondo gorde da”

Beste klase bat sortu “Langile”, aurreko klasetik heredatzen duena.
Gehi propietate bi: Soldata,SegurtasunSoziala. Eta propietate Emaila berriro definitzen du (override), dominioa eskolakoa den kontrolatzeko: value.Substring(value.Length - 14, 14)!= “@iesunibhi.com”. Ez badago ondo, orduan errorea sortu:
Exception ex = new Exception(“Emaila ez duzu ondo jarri"); throw ex; (try, catch, exception egitura erabili errorea kontrolatzeko)
Eta metodo Gorde berriro definitzen du (override), mezua aldatuta “Langilea ondo gorde da”

Sarrera. Idazmahairako interfaze garapena

HERENTZIA eta POLIMORFISMOA

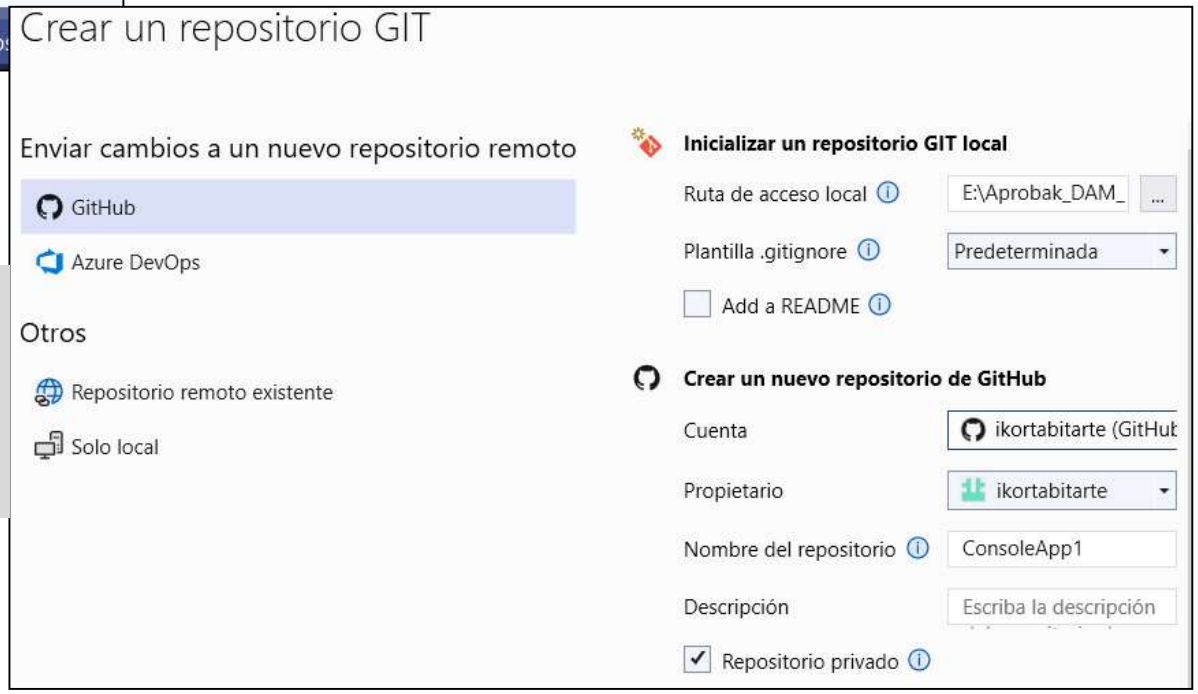
Ariketa 3:



Sarrera. Idazmahairako interfaze garapena

GitHub-era konektatu:

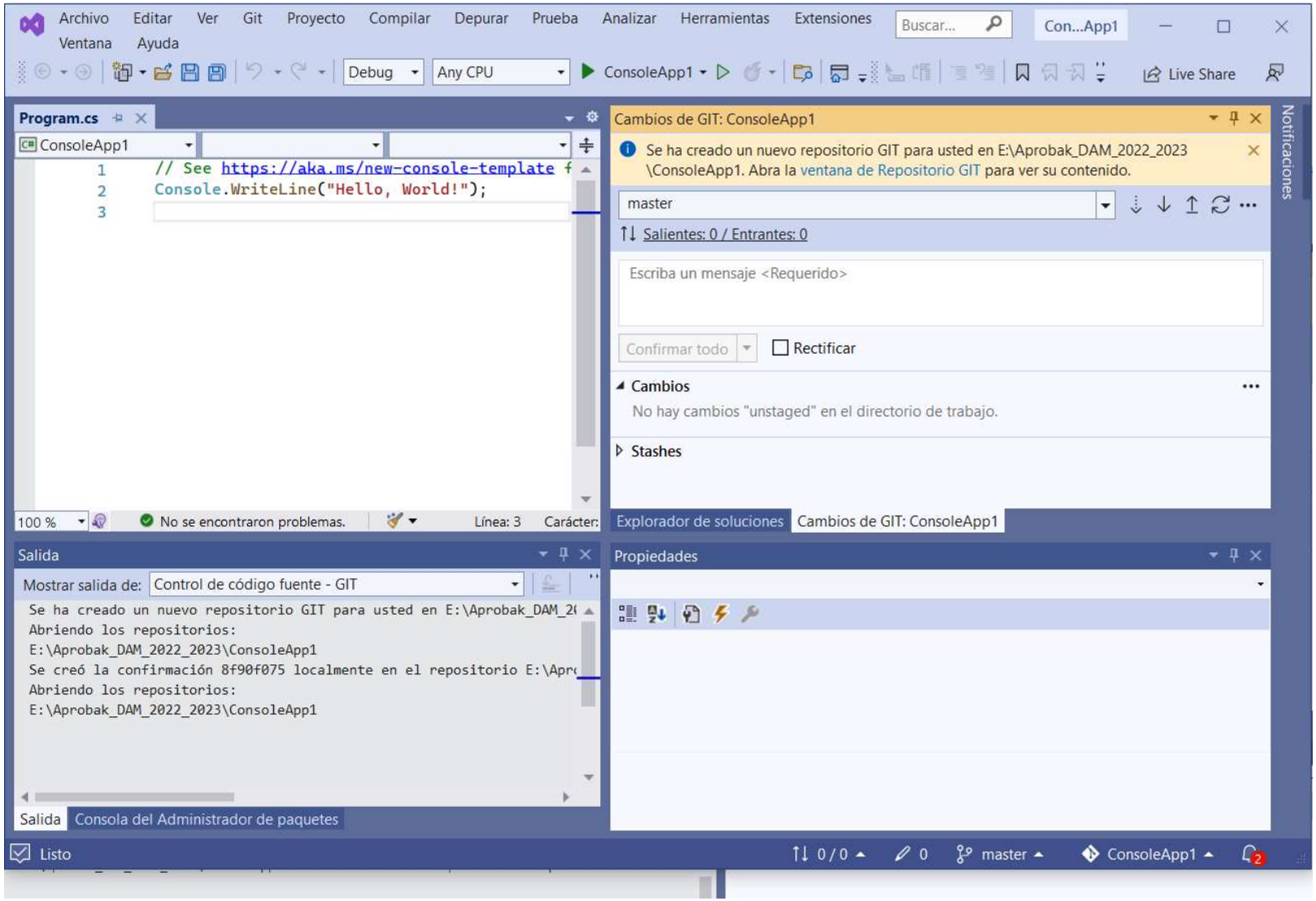
«Crear un nuevo repositorio de GitHub» **aukeratu**



Propietario:
GitHub-en dudan «Organización» bat aukeratu, adibidez:
UNI-DAM2-IG

Sarrera. Idazmahairako interfaze garapena

GitHub-era konektatu:



Sarrera. Idazmahairako interfaze garapena

Estekak:

- Izena aldatu proiektu osoari:

https://medium.com/c-sharp-progarmming/safely-rename-a-project-folder-visual-studio-f3c6bd4d0bd6