Algebra

visoka škola za
primijenjeno računarstvo

OOP - praktikum u .NET okolini
Predavanje 1

www.racunarstvo.hr

O kolegiju

- **Cilj kolegija:**
  - Naučiti razvijati Windows aplikacije u .NET okolini
  - Produbiti znanje OOP-a
- **Kolegij nosi 5 ECTS bodova**
  - 15 sati predavanja
  - 45 sati vježbi
  - 65 sati samostalnog rada

Algebra
visoka škola
za primijenjeno
računarstvo

Uvjeti za potpis

- **Dolasci na predavanja i vježbe**
- **Redovni studenti:**
  - 70% predavanja
  - 80% vježbi
- **Izvanredni studenti:**
  - 60% predavanja i vježbi

Algebra
visoka škola
za primijenjeno
računarstvo

Prikupljanje bodova i ocjene

- **Ukupno 100 bodova:**
  - Prisutnost na nastavi: **3 boda**
  - Međuispiti: **97 bodova**
- **Ocjene:**
  - 92,01 – 100,00 bodova: izvrstan (5)
  - 75,01 – 92,00 bodova: vrlo dobar (4)
  - 58,01 – 75,00 bodova: dobar (3)
  - 50,01 – 58,00 bodova: dovoljan (2)

Razdioba bodova

	M1	M2	Dolaznost	MAX
I1	25			25
I2	24			24
I3		24		24
I4		24		24
Izvan ishoda			3	3
Ukupno	49	48	3	100

Ispiti

- Ispiti i gradivo:
  - Prvi međuispit pokriva ishode učenja 1 i 2
  - Drugi međuispit pokriva ishode učenja 3 i 4
- Ishode učenja je moguće popravljati na svim rokovima

Algebra
visoka škola
za primijenjeno
računarstvo

Windows Forms

- **Windows Forms**
  - Skup klasa, kontrola i alata za razvoj Windows aplikacija
  - Dio .NET programskog okvira
  - Naglasak na brzom razvoju (engl. RAD – *rapid application development*)
- **Tipovi aplikacija:**
  - Desktop
  - Konzolne
  - Web
- **Razvijaju se u Visual Studio-u**

Pomoćne tehnologije

- **U Windows Forms aplikacijama često koristimo:**
  - **ADO.NET/Entity Framework** za pristup izvorima podataka
  - **GDI+** za iscrtavanja i bojenja

- **Alternativa Windows Forms aplikacijama su Windows Presentation Foundation aplikacije (WPF aplikacije)**
  - Naglasak na grafičkim mogućnostima.
  - Umjesto GDI+ za iscrtavanje koriste **DirectX**

Visual Studio

- **Integrirana razvojna platforma**
  - Služi za razvoj svih tipova aplikacija
- **Sadržava niz korisnih alata:**
  - **Vizualne dizajnere** za Windows forme s povuci-pusti (engl. *drag-and-drop*) kontrolama
  - **"Kodno-svjesne" editore** (engl. *code-aware*) koji uključuju kompletiranje naredbi, provjeru sintakse i druge *IntelliSense* mogućnosti
  - **Integrirano prevođenje i otkrivanje grešaka**

Algebra
visoka škola
za primijenjeno
računarstvo

Arhitektura rješenja

- Ishodišni element u izradi aplikacija u Visual Studio-u naziva se rješenje (engl. *Solution*)
- Svako rješenje sadržava jedan ili više projekata (engl. *project*)
- Svaki projekt sadržava razne datoteke i mape, ovisno o veličini i vrsti projekta
- Pretvaranje projekta u izvršni kôd:
  - **Build -> Build Solution** stvara izvršni kôd
  - **Debug -> Start Debugging** stvara izvršni kôd i pokreće aplikaciju u *debug* načinu rada
  - **Debug -> Start Without Debugging** stvara izvršni kôd i pokreće aplikaciju

Dijelovi radne okoline

- Radna okolina Visual Studio-a sastoji se od niza dijelova, od kojih su najbitniji:
  - Središnji dio:
    - Okvir **Design**
    - Okvir **Code**
  - Okvir **Properties** (`Alt + Enter`)
  - Okvir **Solution Explorer** (`Ctrl + Alt + L`)
  - Okvir **Toolbox** (`Ctrl + Alt + X`)

Forme

- **Forme** (engl. *Forms*) su osnovni građevni blokovi korisničkog sučelja u desktop aplikacijama
- Forma predstavlja kontejner koji sadrži kontrole:
  - Omogućuje prezentiranje aplikacije na poznat i konzistentan način
  - Može primiti korisnički unos preko tipkovnice ili miša
  - Može prikazivati podatke u svojim kontrolama
- Jednostavnije aplikacije koriste jednu, a složenije i više formi
- Izgled formi se definira u Design okviru, a instance formi se kreiraju i prikazuju za vrijeme izvršavanja (engl. *run-time*)

Demo

- **Hello, World!**
  - Kreirati jednostavnu Windows Forms aplikaciju i pokrenuti je bez ikakvih izmjena

Algebra
visoka škola
za primijenjeno
računarstvo

Klasa Form

- Sva funkcionalnost formi je implementirana u ugrađenoj klasi `Form`
- Kad dodamo novu formu u projekt, nasljeđujemo tu klasu
  - Klasa je `partial` jer je definirana u dvije datoteke:
    - .Designer.cs koristi dizajner i tu mi ne pišemo ništa
    - .cs koristimo mi i tu dizajner ne piše ništa

Algebra
visoka škola
za primijenjeno
računarstvo

Svojstva Windows formi

- Dostupna su nam brojna svojstva kojima definiramo izgled i ponašanje forme:
  - Pomoću okvira **Properties**
  - U kôdu pomoću **IntelliSense** alata
- Primjerice:
  - Možemo u dizajneru promijeniti naziv forme, podesiti svojstva `Text`, `Size`, `StartPosition`
  - Možemo kroz kôd promijeniti svojstvo `BackColor`
    - Možemo ga podesiti ili na imenovanu boju ili koristiti `Color.FromArgb()`

Rad s kontrolama

- Kontrole dodajemo na formu
  - Iz Toolboxa
  - Kroz kôd
- Forma sve kontrole drži u kolekciji **Controls**
  - sve što je tamo, forma će iscrtati
- Primjerice, možemo na formu dodati dva gumba i podesiti im `Text`, `Size`, `BackColor`, `Location`:
  - Jedan iz Toolboxa
  - Drugi kroz kôd:

```
Button btnZatvori = new Button();
this.Controls.Add(btnZatvori);
```

Algebra
visoka škola
za primijenjeno
računarstvo

Rad s događajima

- Forme i kontrole mogu "ispucati" niz događaja
  - Mi se pretplaćujemo na one koji nas zanimaju
  - Definiramo metode koje želimo da budu pozvane kad se događaj desi
- Primjerice, možemo na oba gumba loviti `Click` događaj:
  - Kroz okvir Properties (gumb Events)
  - Kroz kôd:

```
btnZatvori.Click += btnZatvori_Click;
...
private void btnZatvori_Click(object sender,
EventArgs e)
{
    Application.Exit();
}
```

Algebra
visoka škola
za primijenjeno
računarstvo

Kontejnerske kontrole

- **Kontejnerske kontrole** (engl. *container controls*) sadrže druge kontrole
  - Kontejnerska kontrola može sadržavati i druge kontejnerske kontrole tvoreći tako **hijerarhiju kontrola**
- **Ciljevi:**
  - Korisniku pružiti smisleno korisničko sučelje
  - Lakše upravljanje nad kontrolama sadržanim u njima
- **Primjeri kontejnerskih kontrola:**
  - `Panel`, `GroupBox`, `FlowLayoutPanel`, `TableLayoutPanel`, `TabControl`, `SplitContainer`

Dodavanje kontrola u kontejner

- Kad dodamo kontrolu u neku drugu kontrolu nazivamo je dijete (engl. *child control*), a tu drugu kontrolu roditelj (engl. *parent control*)
- Dva načina dodavanja:
  - U vrijeme dizajniranja
    - *drag-n-drop* na kontejnersku kontrolu
  - U vrijeme izvršavanja
    - Svaka kontejnerska kontrola sadrži svojstvo **`Controls`**
      - Svojstvo `Controls` je kolekcija objekata tipa `Control`
      - Dodavanjem objekta u kolekciju on se automatski prikazuje unutar kontejnerske kontrole.
- Klasa Form također sadržava svojstvo Controls i smatramo je kontejnerskom kontrolom

Svojstva Anchor i Dock

- Jedno bitno svojstvo smo upoznali: **Controls**
- Svojstva Anchor i Dock kažu kako će se djeca ponašati unutar roditelja
  - **`Anchor`** definira udaljenost između jednog/više rubova djeteta i roditelja pri promjeni veličine roditelja
    - Podrazumijevana vrijednost je `Top, Left`
  - **`Dock`** omogućava pričvršćivanje djeteta uz rub roditelja
    - Podrazumijevana vrijednost je `None`
- Primjerice, vežimo oba gumba uz donji desni kut forme

Panel i GroupBox kontrole

- **`Panel`** je osnovna kontejnerska kontrola
  - Najčešće se koristi
  - Predefinirano nema okvir, ali se može postaviti (`BorderStyle`)
- **`GroupBox`**
  - Predefinirano sadrži okvir i tekst koji opisuje grupu
  - Prvenstveno služi za grupiranje `RadioButton` kontrola
    - Unutar jedne `GroupBox` kontrole samo jedna `RadioButton` kontrola može biti odabrana
  - Može se koristiti za grupiranje bilo kakvih kontrola

FlowLayoutPanel kontrola

- Naslijeđena iz `Panel` kontrole
- Dodana dinamička preraspodjela djece u slučaju promjene veličine
  - Princip sličan raspodjeli `HTML` elemenata na web stranici
- Predefinirani smjer toka djece je slijeva nadesno (svojstvo `FlowDirection`)
- Hoće li djeca prijeći u novi redak/stupac određuje svojstvo `WrapContents` (podrazumijevano true)
- Ručni prelazak u novi redak/stupac moguć pomoću `SetFlowBreak()`
  - Čitanje pomoću `GetFlowBreak()`

TableLayoutPanel kontrola

- Predstavlja tablicu
- Svaka ćelija služi kao kontejner za kontrolu
- U ćeliju možemo staviti i kontejnersku kontrolu
- Bitna svojstva:
  - **`CellBorderStyle`** određuje prikaz okvira ćelija
  - **`RowStyles`** i **`ColumnStyles`** predstavljaju kolekcije redaka, odnosno stupaca i preko njih možemo podešavati širinu i visinu
  - **`Controls`** omogućuje dodavanje kontrola:
    - U prvu slobodnu ćeliju prema svojstvu **`GrowStyle`**
    - Točno u određenu ćeliju

TabControl kontrola

- Omogućava grupiranje kontrola na karticama
- TabControl sadržava sljedeća bitna svojsta:
  - **`TabPages`** je kolekcija kontrola tipa `TabPage`
  - **`SelectedIndex`** određuje prikazani `TabPage`
- Svaka kartica je jedna **`TabPage`** kontrola
  - `TabPage` sadržava svojstvo **`Controls`**
- Važniji događaji `TabControl` kontrole:
  - `SelectedIndexChanged`
  - `Selecting`

SplitContainer kontrola

- Sastoji se od razdjelnika (engl. *splitter*) koji razdvaja dvije `SplitterPanel` kontrole
  - `SplitterPanel` kontrola je vrlo slična `Panel` kontroli
    - Dostupne kroz svojstva `Panel1` i `Panel2`
- Bitna svojstva:
  - **`Orientation`** određuje orijentaciju razdjelnika
  - **`FixedPanel`** definira koji panel će ostati fiksiran ako se promijeni veličina cijele kontrole
  - **`IsSplitterFixed`** onemogućuje pomicanje razdjelnika
  - **`SplitterDistance`** postavlja razdjelnik
  - **`SplitterWidth`** određuje širinu/visinu razdjelnika