Jazyk C# 1

10. seminář

Jakub Večeřa

Univerzita Palackého v Olomouci

6. 12. 2024

Reakce na úkoly

- Omlouvám se za neopravení předchozích úkolů
- Tento týden budou všechna řešení opraveny
- Code reviews velice pěkné - většinou jsem byl mírnější
- Dnes bude druhý povinný úkol – potřeba získat alespoň 3 body.

GUI - WPF

- File→New project→WPF Application
- WPF je nástupce WinForms (Win32Api)
- Není multiplatformní ⇒ funguje pouze na Windows
- Kdo nemá Windows - vzdalená plocha: ts.inf.upol.cz
- Přidává definice GUI v XAML (odnož XML)
- Oddělení logiky a vzhledu
- Systém událostí

Designér vzhledu

- „Klikací editor pro desing“
- Možnost cokoliv upravit v XML
- Okno Toolbox – seznam komponent, které lze použít
- Okno Properties – vlastnosti komponent
  - Ikona klíče – vlastnosti komponenty
  - Ikona blesku – události komponenty (dvojklik definice v kódu)

Komponenty

- Checkbox
  - Vlastnost IsChecked
  - Vlastnost ClickMode
- ComboBox
  - Items/ItemsSource
  - SelectedIndex/SelectedItem
- TextBox
  - Vlastnost Text
  - Událost TextChanged

Dialogy

- Jednoduchý oznamovací dialog

```
MessageBoxResult res = MessageBox.Show("Text", "Titulek okna", MessageBoxButton.OK);
```

- Ano/ne dialog

```
MessageBoxResult result = MessageBox.Show("Opravdu odstranit?", "Potvrzení",
MessageBoxButton.YesNo, MessageBoxImage.Question);
if (result == MessageBoxResult.Yes)
{
   ...
} else {
   ...
}
```

Timer

- Může se hodit provádět nějakou akci periodicky
- Třída `DispatcherTimer` v namespace `System.Windows.Threading`
- Událost `Tick` – delegát `Tick(object sender, EventArgs e)`

```
System.Windows.Threading.DispatcherTimer timer = new System.Windo
timer.Tick += dispatcherTimer_Tick;
timer.Interval = new TimeSpan(0, 0, 1);
timer.Start();
```

- Zastavení pomocí `Stop();`

Rozumná architektura – MVVM

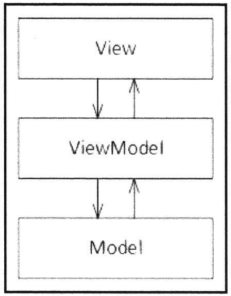
- Živelné „hardcodování“ s ovládacími prvky není dobrý nápad
  - U malých aplikací možná zvládneme
  - Dříve nebo později se stane neúnosným
  - Velmi často zapomeneme „něco někde resetovat“

Rozumná architektura – MVVM

- Živelné „hardcodování“ s ovládacími prvky není dobrý nápad
  - U malých aplikací možná zvládneme
  - Dříve nebo později se stane neúnosným
  - Velmi často zapomeneme „něco někde resetovat“
- Potřeba rozumné architektury, která *rozdělí odpovědnosti*
- Pevně stanovené komponenty, každá vlastní část funkcionality
- Pořádek nás odmění lepší udržitelností projektů

Model-View-ViewModel (MVVM)

- Podobné jako MVC i MVP (rozdělení odpovědností)
- Striktní oddělení uživatelského rozhraní
- Interakce mezi View a ViewModel pomocí *data bindingu*
- Desktopové, mobilní, webové aplikace



Model-View-ViewModel (MVVM)

- *Model* – spravuje aplikační data, udržuje jejich stav, persistence
  - Ale také aplikační logika
- *View* – uživatelské rozhraní
  - Aktivní – řeší si vlastní, události (jak co zobrazit)
  - Neudržuje stav – přeposílá akce ViewModelu
  - Velká část komunikace – *databinding*, málo kódu
- *ViewModel* – veškerá zobrazovací logika
  - Udržuje stav UI
  - „Nezná svůj View“
  - Uživatelské akce překlápí do Modelu

MVVM a WPF – ukázka

- Ukázka MVVM ve WPF na plátně
- Zdrojové kódy budou na webu

Úkol

- Naprogramujte jednoduchý slotový automat

Úkol – specifikace

- Automat bude generovat 3 čísla, zobrazovat level a skóre
- Uživatel bude mít možnost automat zastavit a spustit
- Level bude určovat rychlost generování čísel
- Zastaví-li dvě stejné přičte se mu do skóre jejich součet
- Zastaví-li tři stejné přičte se mu jejich součet $*$ level
- Při *úspěšném* zastavení se zvýší level a rychlost
- Při *neúspěšném* zastavení se mu odečte dvojnásobek násobku hodnot čísel a sníží se level
- Hra bude rozumně informovat uživatele, kterou kombinaci „trefil“ a kolik (ne)získal bodů
- Pohrajte si s funkcionalitou a vzhledem
- Bonusový bod za implementaci pomocí architektury MVVM