

Objektovo Orientované programovanie

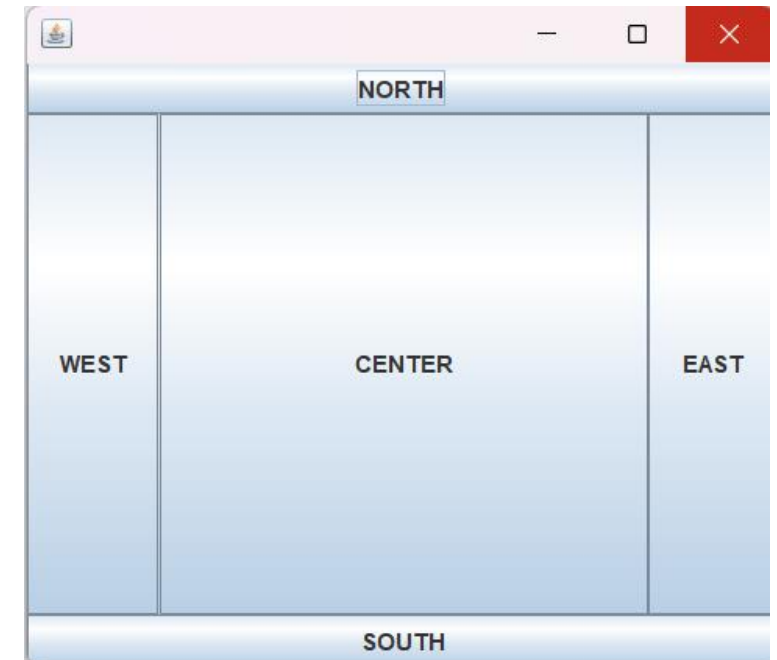
6. Prednáška
LS 2024/2025
Juraj Petrík

Test

- 8.4.2025 – 8:00 – 11:00
- 1. - 7. prednáška a cvičenia
- 20 bodov, 8 minimum

Layout

- Pridávanie widgetov (komponentov) do framu
- Kreslenie 2D grafiky
- Vloženie obrázku



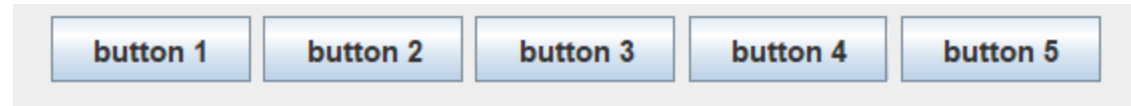
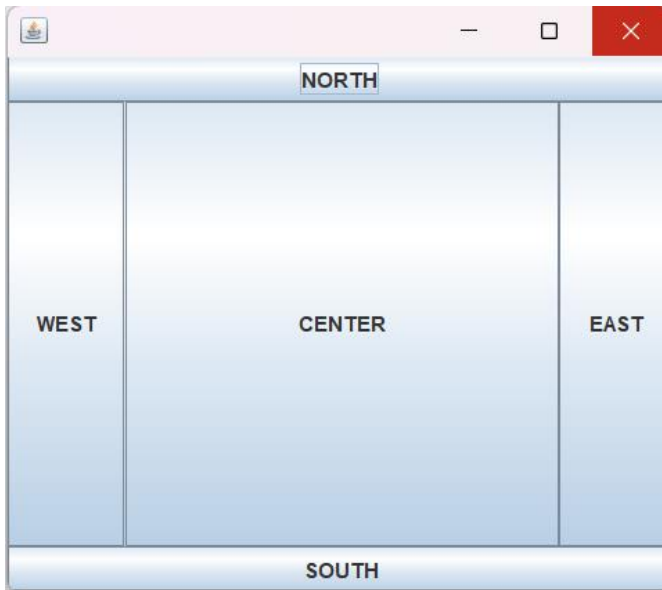
- <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

Layout managers

- Komponenty je štandardné možné vnoriť do seba
- BorderLayout – jeden komponent na región, zvyčajne nerešpektuje rozmery komponentov (default pre frame)
- FlowLayout – zľava doprava, rešpektuje rozmery komponentov, wrapping (default pre panel)
- BoxLayout – podobne ako FlowLayout ale zhora nadol, nema automaticky component wrapping
- GridBagLayout -

Layouts

- Pozor na ďalšie špecifiká, toto je “default” správanie



BorderLayout

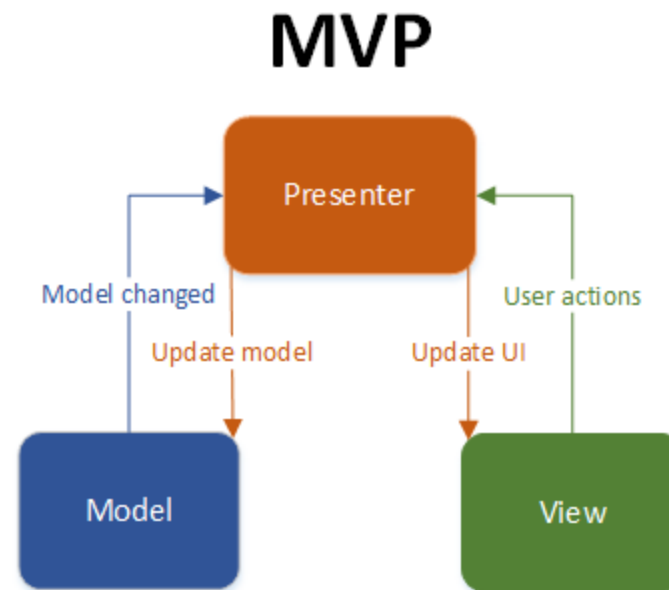
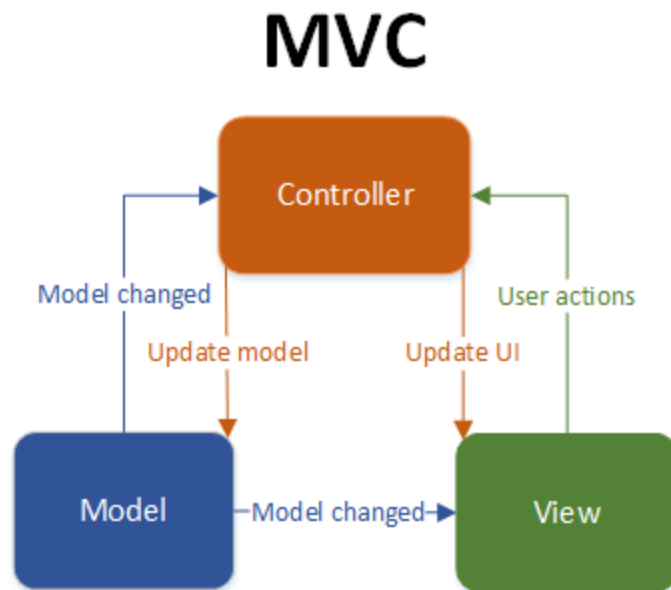
- Ako sa správajú regióny?
- 1 komponent na region
- Ako je to s veľkosťou komponentov?
- It depends..

Komponenty vo Swingu

- Composite
- Background (JFrame, JPanel)
- Interactive components:
 - JButton
 - JLabel
 - JTextField
 - JTextArea
 - JList

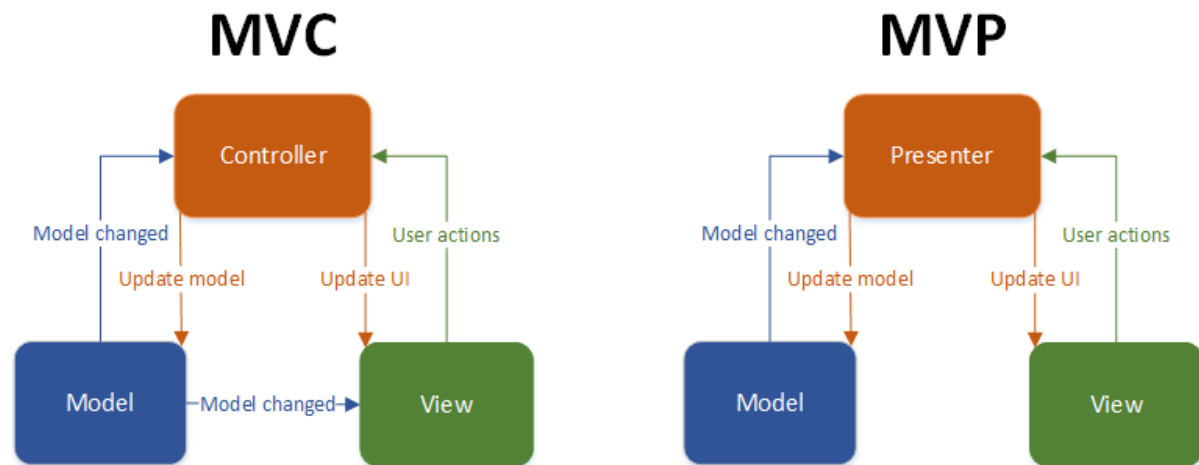
GUI všeobecne

- Čo sa stane keď niečo dlhšie trvá?
- Oddelujte dáta, aplikačnú logiku a prezentačnú vrstvu
- Používajte vhodný vzor: MVC vs MVP vs MVVM vs ...



MVC

- Model reprezentuje samotný dátový model
- View rieši GUI
- Controller rieši „komunikáciu“ medzi modelom a viewom:
 - Implementuje listenery
 - Validuje vstupy
 - Aktualizuje model



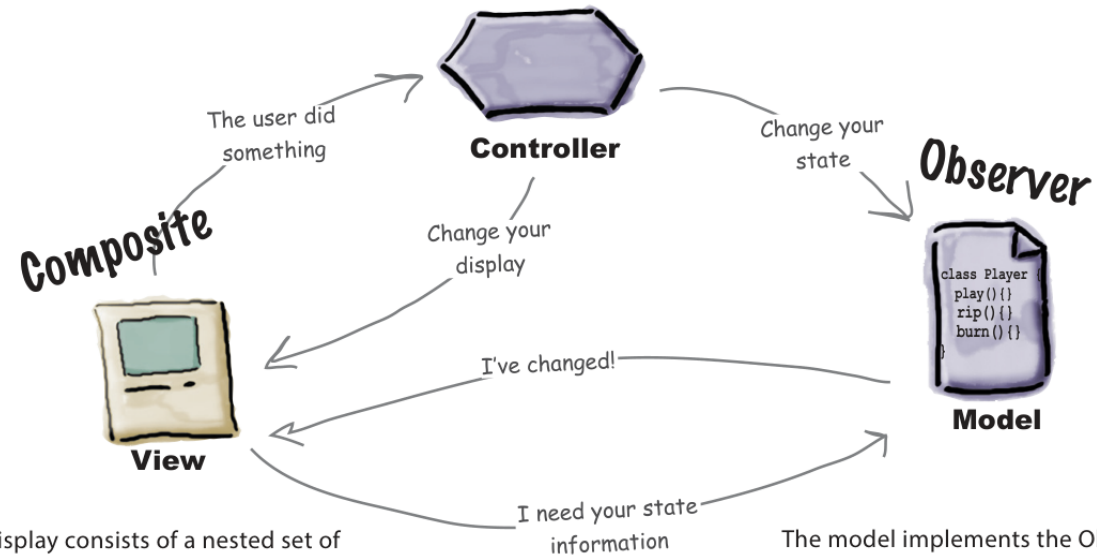
MVC

- View – Composite
- Controller – Strategy
- Model – Observer

- Head First Design Patterns 2nd edition

Strategy

The view and controller implement the classic Strategy Pattern: the view is an object that is configured with a strategy. The controller provides the strategy. The view is concerned only with the visual aspects of the application, and delegates to the controller any decisions about the interface behavior. Using the Strategy Pattern also keeps the view decoupled from the model because it is the controller that is responsible for interacting with the model to carry out user requests. The view knows nothing about how this gets done.

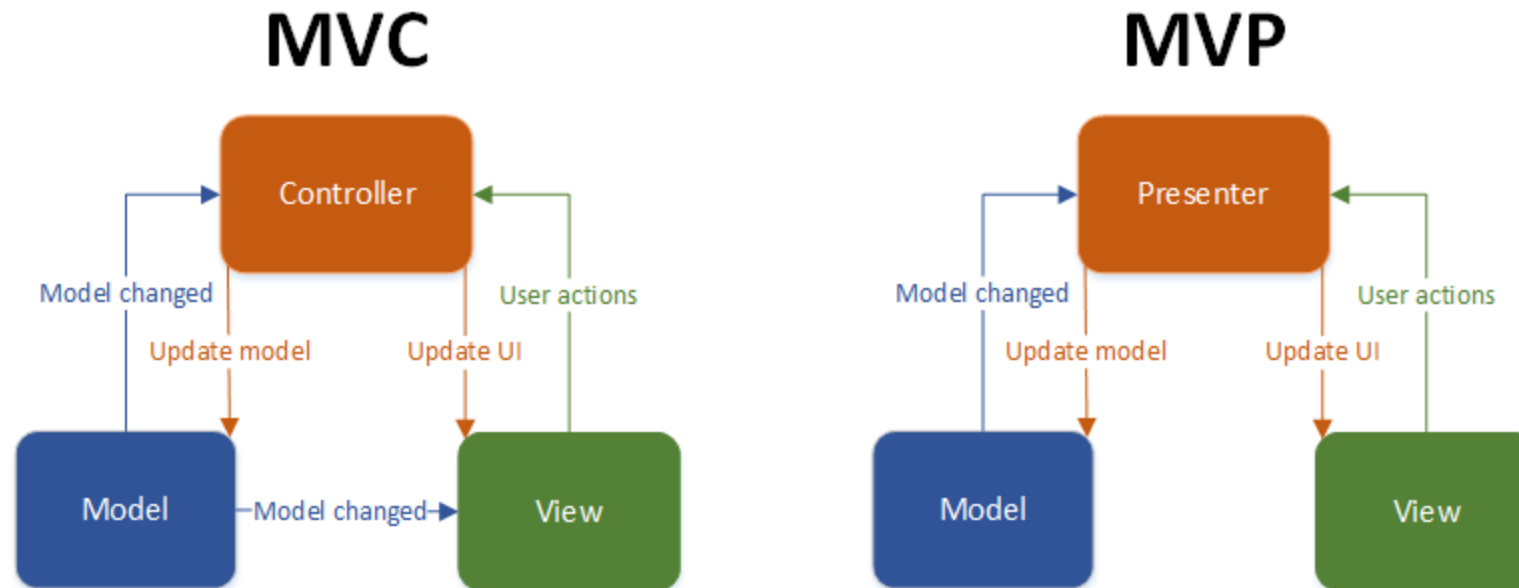


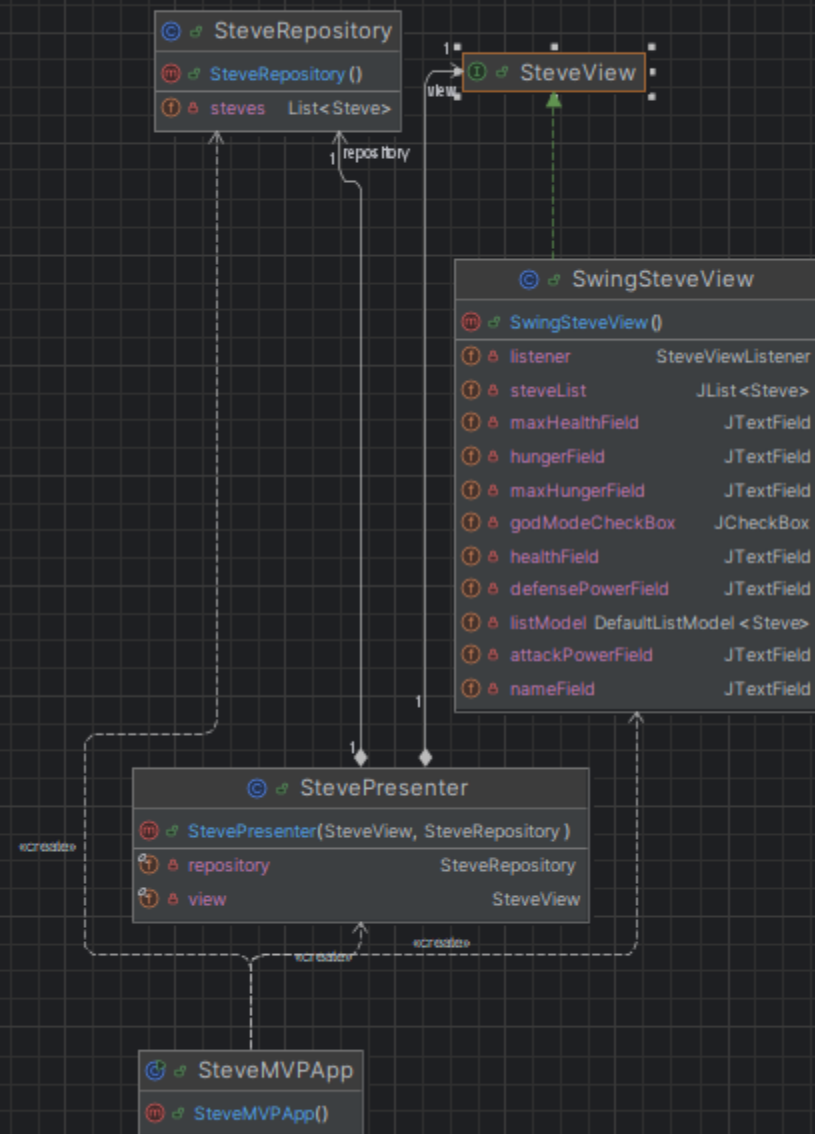
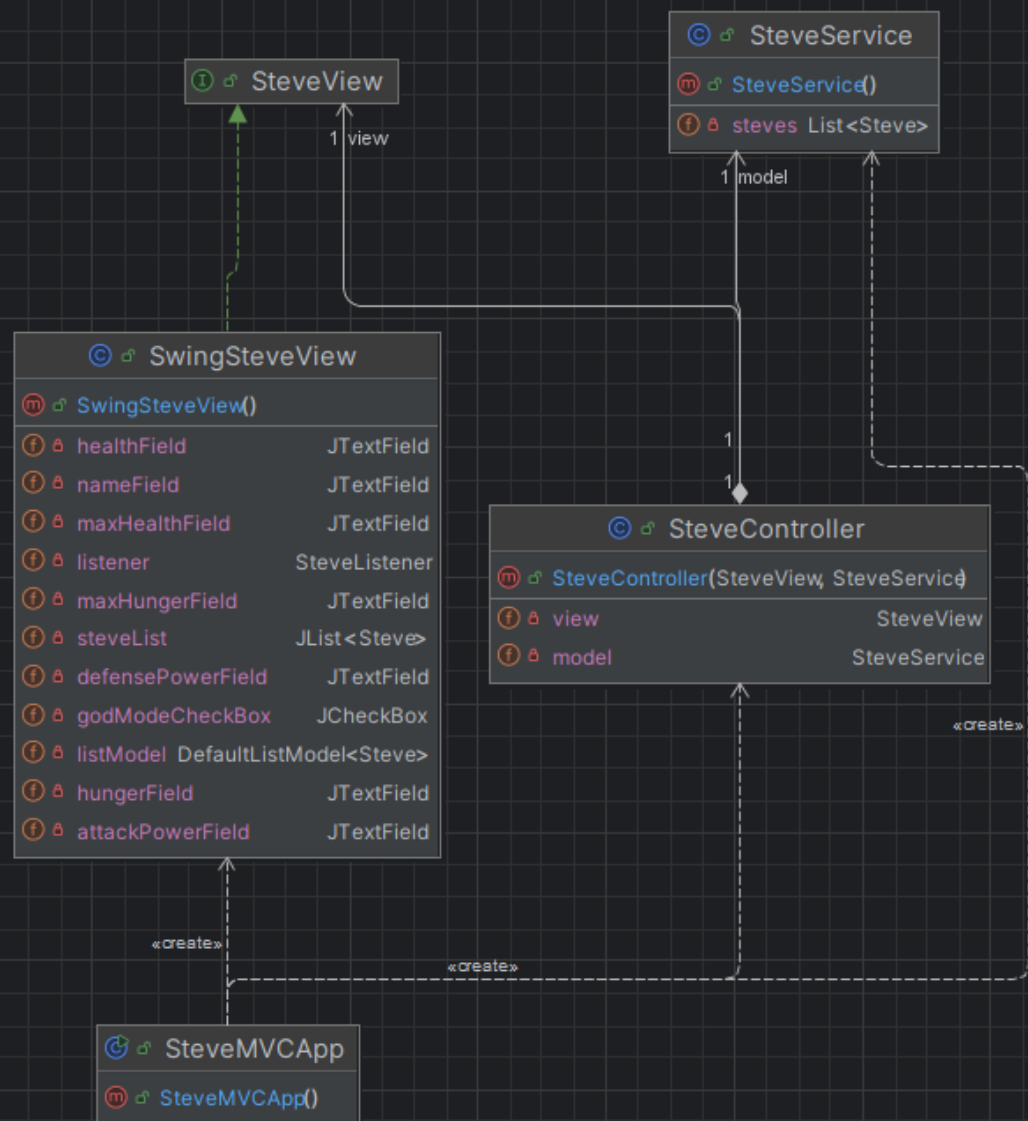
The display consists of a nested set of windows, panels, buttons, text labels, and so on. Each display component is a composite (like a window) or a leaf (like a button). When the controller tells the view to update, it only has to tell the top view component, and Composite takes care of the rest.

The model implements the Observer Pattern to keep interested objects updated when state changes occur. Using the Observer Pattern keeps the model completely independent of the views and controllers. It allows us to use different views with the same model, or even use multiple views at once.

MVP

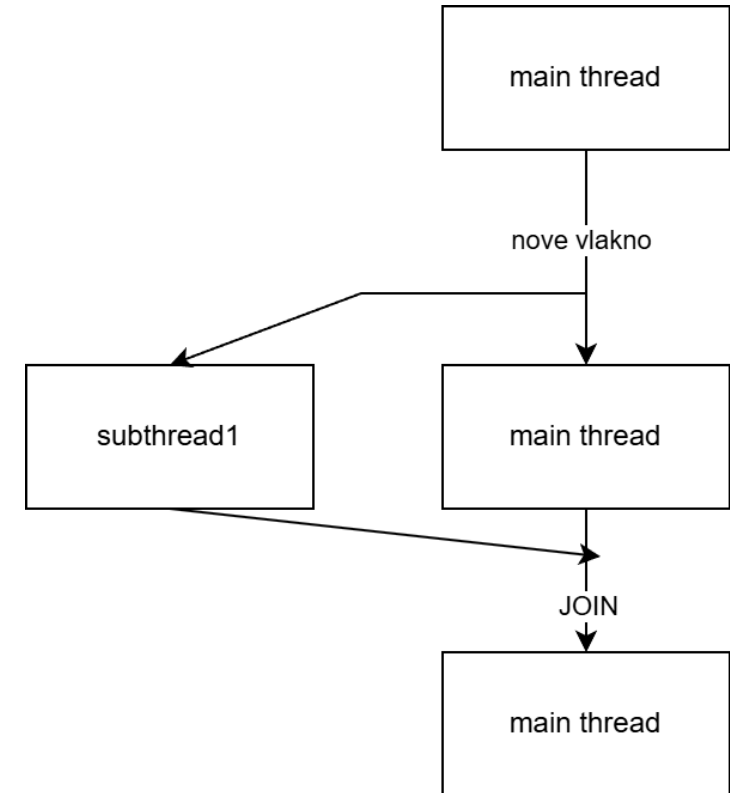
- Model reprezentuje samotný dátový model
- View rieši GUI, „nevie“ o modely
- Presenter je mediátor medzi modelom a viewom





Paralelné spracovávanie pomocou vlákien

- 2 a viac vecí robíme naraz (súbežne)
- Vlákna:
 - “Lightweight procesy”
 - **Spoločná pamäť**
- Pozor na deterministickosť
- Implementujem Runnable
- Swing napr. `javax.swing.SwingUtilities.invokeLater`



Atomickosť

- Synchronized metódy
- V jednom čase iba jedno vlákno smie pracovať s objektom
- Každý objekt má zámok (lock), ale používajú sa iba v prípade ak máme synchronizované časti kódu
- Synchronized na všetko?
 - Deadlock
 - Nevyužívame súbežnosť
- Atomic variables (AtomicInteger) – incrementAndGet, compareAndSet

Immutable objects

- Nie je žiadúce aby sa objekty menili „pod rukami“
 - Final class
 - Final attributes
 - Initialize once
 - No setters!
-
- Pozor na collections, použite vhodnú triedu (napr. `CopyOnWriteArrayList`) – každý proces pracuje nad snapshotom

Runnable vs callable

- Runnable nevracia hodnoty (return) and nevie hodiť checked exception

Quiz time