C# & .NET

http://d3s.mff.cuni.cz

CHARLES UNIVERSITY

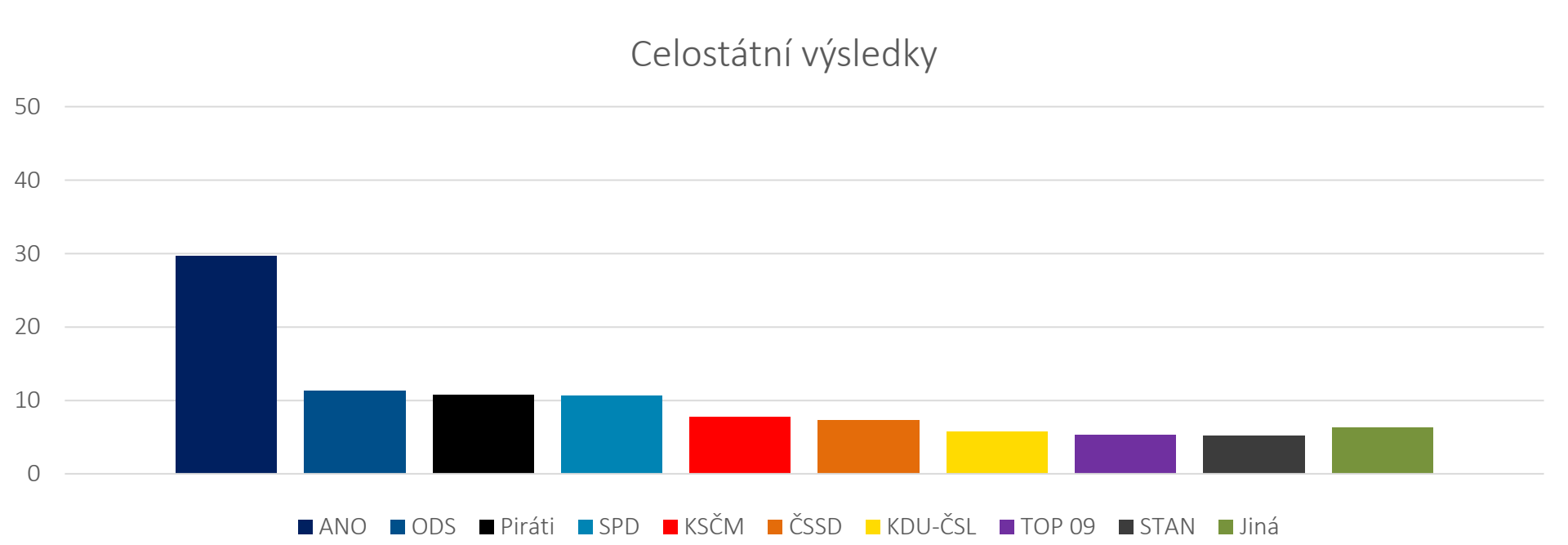
Faculty of Mathematics and Physics

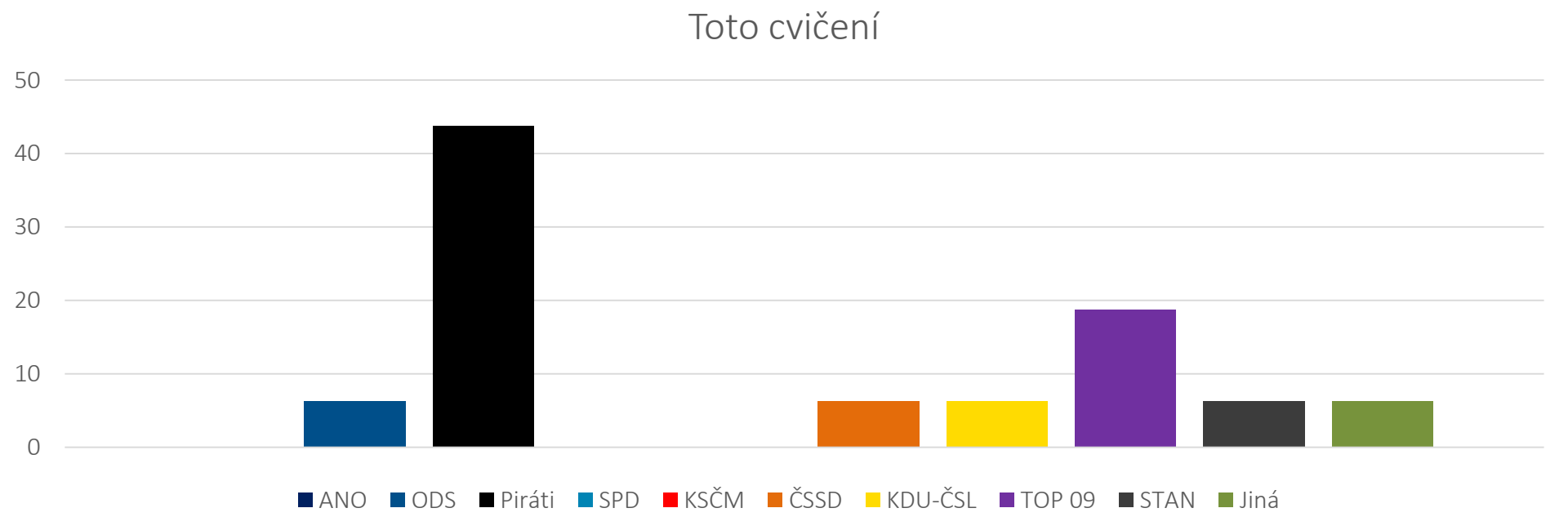
Cvičení

RNDr. Filip Krijt

krijt@d3s.mff.cuni.cz

http://d3s.mff.cuni.cz/~krijt/


Celostátní výsledky
50
40
30
20
10
0
ANO
ODS
Piráti
SPD
KSČM
ČSSD
KDU-ČSL
TOP 09
STAN
Jiná


Toto cvičení
50
40
30
20
10
0
ANO
ODS
Piráti
SPD
KSČM
ČSSD
KDU-ČSL
TOP 09
STAN
Jiná

Plán

- Překvapení z minula
- Objektový návrh
- Nežárka.NET – Rekapitulace
- Kódovací konvence (Coding style)
- Zadání další úlohy: Huffmanovo kódování

Cesta k pěknému kódu

- Přemýšlet v objektech
- Návrh shora dolů
- Best practices
  - Kódovací konvence (Coding Style)
  - Self-documenting code
  - …
- Metodiky, heuristiky
  - TDD – Test Driven Development
  - SOLID – Pět základních principů pro OOP
  - …
- Návrhové vzory

Objektový návrh – SOLID

- Single responsibility principle – třída by neměla dělat / zastřešovat víc jak jednu věc
- Open for extension, closed for modification – mělo by jednoduše jít přidat novou funkcionalitu aniž by se změna propagovala celým programem
- Liskov substitution – design by contract, potomek by měl všude jít použít místo předka
- Interface segregation – preferujeme granulární interfaces (nepřehánět)
- Dependency inversion – spoléhejte na abstrakce (rozhraní), ne konkrétní třídy a implementace

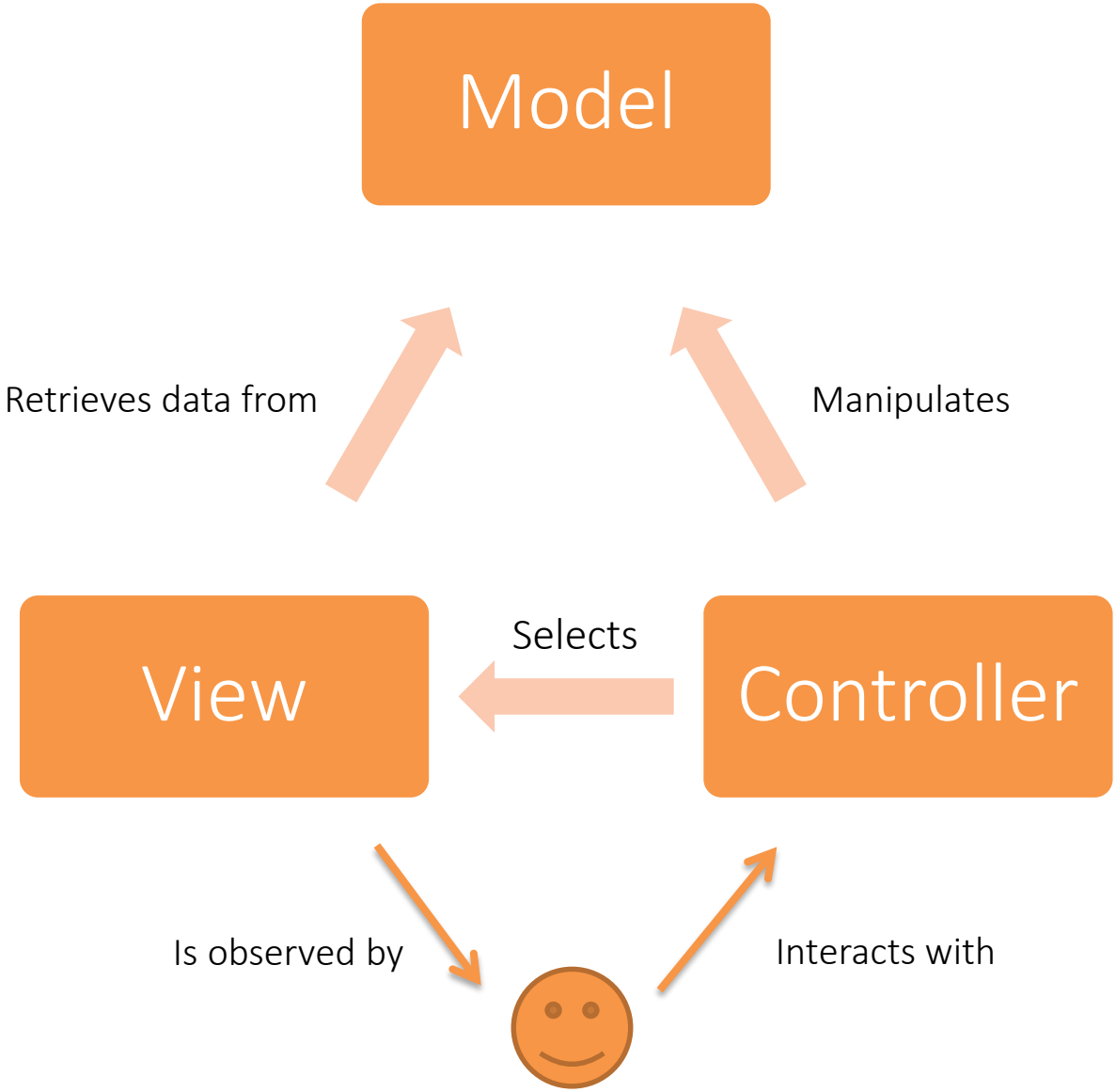
Objektový návrh – heuristiky

- Hledejte kusy funkcionality, které jde seskupit dohromady pod nějaké rozhraní
- Pozor na dlouhé metody a zanoření bloků
  - => Rozbít na menší kusy
- Pozor na duplikaci kódu
  - => Extrakce nové metody, třídy, nebo konstanty
- Pozor na globální prvky (static)
  - V principu není příliš objektové
  - Main musí být, ale jinak minimum
- Vizuální rozdělení věcí pomocí konců řádků
  - Odřádkování za metodou, blokem atp.

Objektový návrh – heuristiky

- Pozor na omnitřídy
  - Třída Reader by asi neměla mít metodu WriteLine :)
  - Reader a Writer v jednom
  - => Single responsibility principle, dvě třídy
- Pozor na nadměrnou delegaci
  - Main(args) { Run(args) }
- Cílem je vždycky zajistit nějaké kvalitativní vlastnosti => Analyzovat, zda skutečně návrhem něco zajišťuji, nebo jen zbytečně komplikuji

MVC design pattern



Různé varianty oddělení funkcionality

- Na úrovni metod – většina
- Na úrovni tříd
  - Skládáním – oficiální řešení
  - Dědičností
- Proč má v případě Views smysl rozdělení do více tříd?
  - Lepší testovatelnost a rozšiřitelnost
  - Lepší sémantika – jeden View odpovídá konkrétní webové stránce
  - Funkcionalita se zapouzdří do dat => větší flexibilita

Zajímavé otázky

- Kam umístit přidání a odebrání knihy?
- Má smysl mít IModel, IView a IController?
- Má smysl mít více Controllerů?
- Má smysl reprezentovat požadavky samostatnými objekty?
- Měl by být Controller bezstavový?
- Kdo by měl vytvářet konkrétní views (ve smyslu volání new)?

Stringy a řízení programu

- Stringy jsou vhodné pro textová data na zobrazení uživateli, ale máloco jiného
- Propagaci stringů programem je vhodné utnout co nejdříve to jde
  - Tedy např. mít jasně oddělené parsování vstupu a jeho interpretaci
- Na předávání rozumně omezené množiny příkazů dál do programu je vhodnější výčtový typ (enum)
  - Hlavně kvůli kontrole kompilátorem
  - Enum se hodí i místo intových identifikátorů operací
- Pro složitější případy např. návrhový vzor Command