CSharpPlayground – Zadání (rozšířená verze)

*Konzolová aplikace v C#/.NET 8 pro procvičení klíčových konceptů jazyka a ekosystému.*

# Obsah

* Modul 1: Hlavní menu
* Modul 2: Kolekce & LINQ
* Modul 3: OOP & SOLID
* Modul 4: Soubory & JSON
* Modul 5: Async & zrušení úloh
* Modul 6: Delegates & Events
* Modul 7: Pattern Matching & Records
* Modul 8: Základní DI (ruční kompozice)
* Modul 9: Chyby & výjimky
* Modul 10: Testy (xUnit)
* Bonusové úkoly
* Akceptační kritéria

# Modul 1: Hlavní menu

**Cíl:**

* Vytvořit hlavní orchestraci aplikace bez business logiky.
* Oddělit UI (konzoli) od logiky modulů.

**Uživatelský průběh:**

1. Po spuštění aplikace se zobrazí očíslované menu modulů a příkazů (q/exit).
2. Uživatel zadá číslo modulu (1–10) nebo 'q' pro ukončení.
3. Po dokončení modulu se zobrazí hláška a návrat do hlavního menu.

**Úkoly k implementaci:**

* Implementovat smyčku menu (while) s bezpečným čtením vstupu.
* Mapovat volbu → spuštění třídy daného modulu (např. IModule.Run()).
* Zajistit, že selhání modulu neukončí aplikaci (centrální exception handling).
* Přidat volbu 'help' pro stručný popis modulů.

**Vstupy:**

* Volby z klávesnice (číslo, 'q', 'help').

**Výstupy:**

* Čitelný seznam modulů a jejich krátký popis.
* Zpětné potvrzení zvolené akce.

**Edge cases:**

* Neplatná volba (písmena místo čísla, mimo rozsah).
* Neočekávaná výjimka z modulu – zachytit, zalogovat, pokračovat.

**Akceptační testy:**

1. Menu funguje opakovaně; aplikace se ukončí až po 'q'/'exit'.
2. Help vypíše popisy všech modulů.

# Modul 2: Kolekce & LINQ

**Cíl:**

* Procvičit LINQ nad kolekcemi (filtr, projekce, řazení, agregace).
* Oddělit čistou logiku od UI (metody bez I/O).

**Uživatelský průběh:**

1. Uživatel načte předpřipravený seznam produktů (in-memory nebo z JSON – viz modul 4).
2. Vybere operaci: vyhledat / filtrovat / řadit / agregovat.
3. Zadá parametry (např. kategorie='Books', skladem=true, řazení='price desc').
4. Výsledek se vypíše tabulkově v konzoli.

**Úkoly k implementaci:**

* Definovat record Product(id, name, category, price, inStock).
* Implementovat čisté metody: Search, Filter, Sort, Aggregate.
* Vytvořit utilitu pro tisk výsledků v tabulce (zarovnání sloupců).
* Dodat ukázkové datové sady (10–20 produktů).

**Vstupy:**

* Text vyhledávání (case-insensitive).
* Kategorie (více hodnot oddělených čárkou).
* Filtr skladem ano/ne.
* Řazení: 'name asc|desc', 'price asc|desc'.

**Výstupy:**

* Tabulkový výpis vyhovujících položek.
* Agregace: počet, průměrná cena, min/max, grouping podle kategorie.

**Edge cases:**

* Prázdný výsledek – hezká zpráva 'Nic nenalezeno'.
* Neplatné zadání řazení – vrátit nápovědu a vyžádat opakování.

**Akceptační testy:**

1. Všechny operace vracejí správné výsledky pro testovací data.
2. Výstup je čitelný a konzistentní (stejné šířky sloupců).

# Modul 3: OOP & SOLID

**Cíl:**

* Procvičit rozhraní, dědičnost a principy SOLID (zejména DIP).
* Ukázat oddělení domény (služby) od implementací.

**Uživatelský průběh:**

1. Uživatel vytvoří objednávku nad seznamem produktů.
2. Systém vypočte cenu přes vybranou pricing strategii a ověří sklad.
3. Uživatel obdrží rekapitulaci objednávky.

**Úkoly k implementaci:**

* Definovat IPricingStrategy + implementace: FlatDiscount(%) a CategoryBasedDiscount(map<Category, %>).
* Definovat IInventoryService (Check/Reserve/Release).
* Definovat IOrderService, které orchestrace vytváření objednávky.
* Udělat entity: Order, OrderItem (recordy), případně enum OrderStatus.

**Vstupy:**

* Seznam položek košíku (productId, qty).
* Volba strategie cenotvorby.

**Výstupy:**

* Vypočtená celková cena, rozpis slev.
* Stav rezervace skladu a výsledná rekapitulace objednávky.

**Edge cases:**

* Nedostatečné množství na skladě → OutOfStockException.
* Neznámé productId → validační chyba.

**Akceptační testy:**

1. Obě strategie dávají očekávané výsledky (ověřeno testy).
2. Rezervace skladu se provádí idempotentně (opakované volání bezpečné).

# Modul 4: Soubory & JSON

**Cíl:**

* Načítat a ukládat data ve formátu JSON (System.Text.Json).
* Bezpečně ošetřit chyby a validovat data.

**Uživatelský průběh:**

1. Uživatel zvolí 'Načíst produkty ze souboru'.
2. Aplikace načte Data/products.json do modelů.
3. Po filtrování/řazení lze exportovat výsledek do exports/filtered-YYYYMMDD.json.

**Úkoly k implementaci:**

* Vytvořit IJsonStorage<T> s implementací FileJsonStorage<T>.
* Implementovat validaci (např. povinná pole, kladná cena).
* Vytvořit vlastní DataFileException a smysluplné chyby pro uživatele.

**Vstupy:**

* Cesta k souboru (default Data/products.json).
* Volba exportu a cílová cesta (default exports/...).

**Výstupy:**

* Načtený seznam produktů (log počtu načtených záznamů).
* Uložený JSON soubor s výsledky operací.

**Edge cases:**

* Neexistující soubor / neplatný JSON – zachytit a popsat.
* Duplicitní ID produktů – rozhodnout strategii (ignorovat/overwrite/chyba).

**Akceptační testy:**

1. Korektní načtení a uložení, chyby mají srozumitelné hlášky.
2. Serializace/deserializace zvládne i rozšířené vlastnosti (nezlomí se).

# Modul 5: Async & zrušení úloh

**Cíl:**

* Procvičit async/await, CancellationToken a reporting průběhu.
* Naučit se bezpečně ukončovat běžící úlohy.

**Uživatelský průběh:**

1. Uživatel spustí 'Přepočet ceníku'.
2. Během běhu může zadat 'cancel' pro zrušení.
3. Po dokončení se vypíše statistika (trvání, počet zpracovaných položek).

**Úkoly k implementaci:**

* Implementovat asynchronní metodu RecalculatePricesAsync(products, token, IProgress<int>).
* V pravidelných krocích hlásit progress (0–100%).
* Správně reagovat na token.ThrowIfCancellationRequested().

**Vstupy:**

* Parciální velikost šarže (batch size).
* Vyvolání zrušení (uživatelský příkaz).

**Výstupy:**

* Průběžný progress v procentech.
* Souhrn po dokončení/zrušení.

**Edge cases:**

* Zrušení uprostřed souborových operací – rollback/žádná částečná persistace.
* Výjimky uvnitř smyčky – log, pokračování nebo graceful stop.

**Akceptační testy:**

1. Zrušení funguje okamžitě a nezanechává nekonzistentní stav.
2. Progress je monotónní a končí na 100 % při úspěchu.

# Modul 6: Delegates & Events

**Cíl:**

* Pochopit delegáty (Action/Func) a události (event).
* Ukázat publish/subscribe bez těsného propojení.

**Uživatelský průběh:**

1. Po vytvoření objednávky se vyvolá OrderCreated event.
2. Subscribed služba (NotificationService) na to reaguje a vypíše zprávu.

**Úkoly k implementaci:**

* V entitě/doméně definovat event EventHandler<OrderCreatedEventArgs>.
* V NotificationService se přihlásit k odběru.
* Ukázat i malou util metodu, která přijme Func<T,T> (např. transformace).

**Vstupy:**

* Žádné speciální – akci spustí modul 3.

**Výstupy:**

* Záznam o tom, že byla odeslána notifikace (konzole/log).

**Edge cases:**

* Žádný subscriber – event nesmí padat.
* Více subscriberů – pořadí není garantované, ošetřit výjimky.

**Akceptační testy:**

1. Event se vyvolá a alespoň jeden subscriber je zavolán.
2. Chyba v jednom subscriberu nesmí položit aplikaci.

# Modul 7: Pattern Matching & Records

**Cíl:**

* Použít records pro imutabilní datové typy a pattern matching ve switch expression.
* Procvičit dědičnost recordů.

**Uživatelský průběh:**

1. Uživatel zvolí produkt a aplikace vypíše způsob dodání.
2. DigitalProduct → 'download link', fyzický Product → dodací lhůta dle inStock.

**Úkoly k implementaci:**

* Definovat record Product a record DigitalProduct : Product.
* Implementovat GetDeliveryInfo(Product) pomocí switch expression + property patterns.

**Vstupy:**

* ID produktu z dostupného seznamu.

**Výstupy:**

* Krátká textová informace o způsobu dodání.

**Edge cases:**

* Neznámý typ produktu – default větev.
* Null reference – ochrana pomocí nullable a guard clauses.

**Akceptační testy:**

1. Funkce vrací správné texty pro oba typy produktů.
2. Testy pokrývají všechny větve switch expression.

# Modul 8: Základní DI (ruční kompozice)

**Cíl:**

* Sestavit graf objektů na jednom místě bez IoC kontejneru.
* Závislosti předávat konstruktorovou injekcí.

**Uživatelský průběh:**

1. Při startu aplikace se vytvoří instance služeb a předá se jim konfigurace.
2. Moduly dostanou své závislosti přes konstruktory.

**Úkoly k implementaci:**

* Vytvořit třídu App (kompozice) – zde jedině používat 'new' k sestavení služeb.
* Nikde jinde ve třídách nevytvářet závislosti přímo.

**Vstupy:**

* Konfigurace z appsettings.json (viz modul 4).

**Výstupy:**

* Žádné přímé – jde o architektonické pravidlo.

**Edge cases:**

* Kruhové závislosti – refaktor do menších rozhraní.
* Příliš mnoho parametrů v konstruktoru – zavést továrnu/fasádu.

**Akceptační testy:**

1. Všechny služby dostávají závislosti přes konstruktor.
2. App.cs je jediné místo pro instanciace.

# Modul 9: Chyby & výjimky

**Cíl:**

* Navrhnout smysluplnou hierarchii výjimek a centrální zpracování.
* Vytvářet uživatelsky srozumitelné chybové zprávy.

**Uživatelský průběh:**

1. Při chybě se zobrazí přátelská hláška a detail se zaloguje.
2. Aplikace pokračuje (návrat do menu).

**Úkoly k implementaci:**

* Vytvořit DataFileException a OutOfStockException (vlastní message + metadata).
* Globální handler v App – try/catch kolem spouštění modulů.
* Jednoduchý ILogger + ConsoleLogger (čas, typ, message).

**Vstupy:**

* Chybové situace simulované uživatelskými volbami (např. špatná cesta).

**Výstupy:**

* Uživatelská hláška + záznam v logu.

**Edge cases:**

* Nezachycená výjimka – aplikace nesmí spadnout bez vysvětlení.
* Vícenásobné logování stejné chyby – deduplikace ID.

**Akceptační testy:**

1. Běžné chyby jsou zachycené a popsané.
2. Log obsahuje čas, typ a detail chyby.

# Modul 10: Testy (xUnit)

**Cíl:**

* Pokrytí klíčové logiky unit testy a jednoho integračního toku.
* Ukázat testy čistých funkcí i služeb se závislostmi (přes rozhraní).

**Uživatelský průběh:**

1. Spuštění testů přes 'dotnet test'.
2. Vývojář vidí červenou/zelenou (TDD volitelné).

**Úkoly k implementaci:**

* Unit testy: pricing strategie, inventory, LINQ funkce (Search/Filter/Sort).
* Integrační test: OrderService s fake implementacemi rozhraní (bez I/O).
* Testy chybových stavů (výjimky).

**Vstupy:**

* Předdefinované testovací sady dat.

**Výstupy:**

* Zpráva z test runneru (vše zelené).

**Edge cases:**

* Hranové hodnoty (0 qty, záporná cena – pokud validace povolí).
* Race conditions v async – použít deterministický scénář.

**Akceptační testy:**

1. Min. 6–8 unit testů + 1 integrační test prochází.
2. Testy jsou čitelné a nezávislé (Arrange-Act-Assert).

# Bonusové úkoly

* BenchmarkDotNet – porovnání dvou implementací filtrování.
* Serilog – strukturované logování do souboru.
* Polymorfní JSON – diskriminátor typu pro Product/DigitalProduct.
* Extension metody – např. ToTableString() pro kolekce.
* Reflection – dynamické načtení modulů implementujících IModule.
* Nullable reference types – zapnout a opravit warningy.
* Generics – GenericRepository<T> s in-memory úložištěm.

# Akceptační kritéria (revidovaná)

* Aplikace spustí menu a zvládne všechny moduly 2–7 dle specifikace.
* Ruční DI kompozice pouze v App.cs, ostatní třídy pouze přijímají rozhraní.
* JSON načtení/uložení + chybové stavy jsou uživatelsky srozumitelné.
* Async modul podporuje zrušení a reportuje průběh.
* Event OrderCreated vyvolán a zpracován alespoň jedním subscriberem.
* Min. 8 unit testů + 1 integrační test prochází a jsou srozumitelné.