

SEMESTRÁLNÍ PRÁCE A

Maximální možný bodový zisk: **4 body**

A) Motivační příklad:

V rámci výroby nějakého produktu se jeho výroba skládá z jednotlivých procesů. Každý proces může být realizován (i) robotem, nebo (ii) manuálně. Čas od času může dojít k reorganizaci výrobních **manuálních** procesů a to tak, že buď jsou (i) časově náročné procesy dekomponovány, nebo (ii) dva krátké po sobě jdoucí procesy agregovány.

B) Použité datové struktury:

V rámci modulu **ABSTRDOUBLELIST** implementujte abstraktní datovou strukturu (ADS) **obousměrně cyklicky zřetězený lineární seznam** v dynamické paměti (stylizovaně znázorněný v rámci obr. 1). Tato třída implementuje rozhraní `IAbstrDoubleList`, které implementuje implicitní rozhraní `Iterable`. Rozhraní `IAbstrDoubleList` je definováno následovně:

`void zrus()` –zrušení celého seznamu,

`boolean jePrazdny()` –test naplněnosti seznamu,

`void vlozPrvni(T data)` –vložení prvku do seznamu na první místo

`void vlozPosledni(T data)` –vložení prvku do seznamu na poslední místo,

`void vlozNaslednika(T data)` –vložení prvku do seznamu jakožto následníka aktuálního prvku,

`void vlozPredchudce(T data)` –vložení prvku do seznamu jakožto předchůdce aktuálního prvku,

`T zpristupniAktualni()` –zpřístupnění aktuálního prvku seznamu,

`T zpristupniPrvni()` –zpřístupnění prvního prvku seznamu,

`T zpristupniPosledni()` –zpřístupnění posledního prvku seznamu,

`T zpristupniNaslednika()` –zpřístupnění následníka aktuálního prvku,

`T zpristupniPredchudce()` –zpřístupnění předchůdce aktuálního prvku,

Pozn. Operace typu `zpristupni`, přenastavují pozici aktuálního prvku

`T odeberAktualni()` –odebrání (vyjmutí) aktuálního prvku ze seznamu poté je aktuální prvek nastaven na první prvek

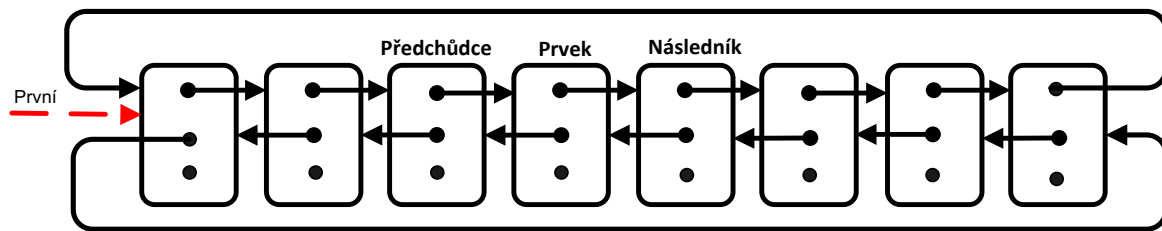
`T odeberPrvni()` –odebrání prvního prvku ze seznamu,

`T odeberPosledni()` –odebrání posledního prvku ze seznamu,

`T odeberNaslednika()` –odebrání následníka aktuálního prvku ze seznamu,

`T odeberPredchudce()` –odebrání předchůdce aktuálního prvku ze seznamu,

`Iterator<T> iterator()` –vytvoří iterátor (dle rozhraní `Iterable`)



Obr. 1: Obousměrně cyklicky zřetěžený lineární seznam

Dále **pomocí** datové struktury **ABSTRDOUBLELIST** vybudujte abstraktní datovou strukturu **zásobník**, jež bude v samostatném modulu **ABSTRLIFO**. Tato datová struktura implementuje následující rozhraní: **IAbstrLifo**

`void zrus()` – zrušení celého zásobníku
`boolean jePrazdny()` – test naplněnosti zásobníku

`void vloz(T data)` – vložení prvku do zásobníku
`T odeber()` – odebrání prvku ze zásobníku

Abstraktní lineární seznam slouží pro uchovávání jednotlivých výrobních procesů. Abstraktní datový typ zásobník pak aplikace využívá pro uchovávání procesů, které mají být reorganizovány (dekomponovány, respektive agregovány).

- C) Pro ověření funkčnosti implementovaných ADS vytvořte modul **VyrobníProces**. Tento modul umožňuje správu výrobních procesů a implementuje následující rozhraní:

`int importDat(String soubor)` – provede import dat z datového souboru `import.csv`. Návrátová hodnota představuje počet úspěšně načtených záznamů.

`void vlozProces(Proces proces, enumPozice pozice)` – vloží nový proces do seznamu procesů na příslušnou pozici (první, poslední, předchůdce, následník)

`Proces zpristupniProces(enumPozice pozice)` – zpřístupní proces z požadované pozice (první, poslední, předchůdce, následník, aktuální)

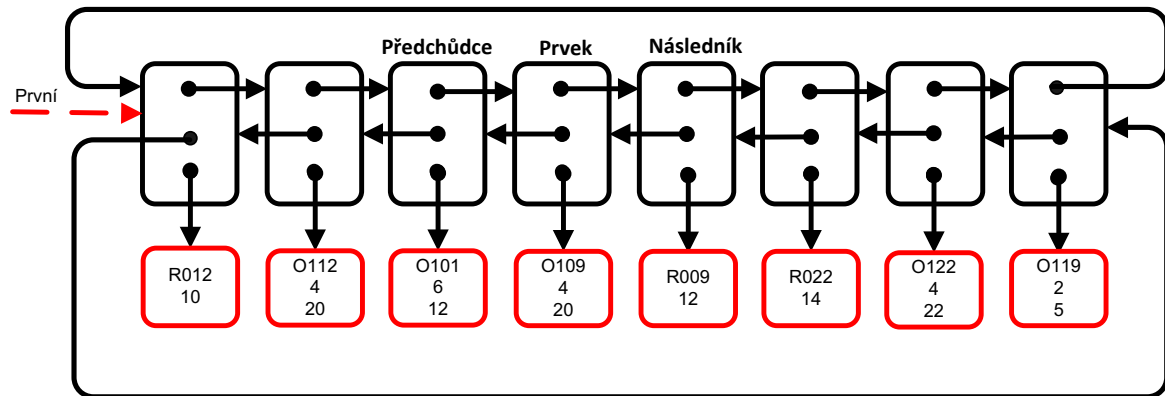
`Proces odeberProces(enumPozice pozice)` – odebere proces z požadované pozice (první, poslední, předchůdce, následník, aktuální)

`Iterator iterator()` – vrátí iterátor

`IAbstrLifo vytipujKandidatiReorg(int cas, enumReorg reorgan)` – pomocí iterátoru provede vytipování všech procesů k reorganizaci (dekompozice/agregace). Vstupním kritériem je čas a typ potenciální reorganizace. Vytipování kandidáti jsou ukládání do zásobníku.

`void reorganizace(enumReorg reorgan, IAbstrLifo zasobnik)` – s vybranými procesy provede požadovaný typ reorganizace (dekompozice/agregace)

`void zrus()` – zruší všechny procesy.



Obr. 2: Stylizované znázornění procesů výroby

Modul **Procesy** – třídy procesů jsou potomci abstraktní třídy **Proces**

- **ProcesRoboticky**
 - i. id
 - ii. casProcesu
- **ProcesManualni**
 - i. id
 - ii. pocetOsob
 - iii. casProcesu

D) Pro obsluhu aplikace vytvořte uživatelské **formulářové** rozhraní **ProgVyrobníProces**, které umožňuje obsluhu programu a volat požadované operace.

Zmíněný program, necht' umožňuje zadávání vstupních dat z klávesnice, ze souboru a z generátoru, výstupy z programu necht' je možné zobrazit na obrazovce a uložit do souboru.