SEMESTRÁLNÍ PRÁCE A

Maximální možný bodový zisk: **4 body**

A) Motivační příklad:
V rámci agendy autopůjčovny jsou vedeny jednotlivé pobočky s možností zapůjčení automobilu. Každá pobočka disponuje seznamem aut, která si na dané pobočce lze zapůjčit. Vypůjčený automobil je odebrán z nabídky, a je následně evidován/vložen v seznamu vypůjčených aut. Po navrácení vypůjčeného vozu je aktualizován stav ujetých kilometrů a počet výpůjček.

B) Použité datové struktury:
V rámci modulu **ABSTRDOUBLELIST** implementujte abstraktní datovou strukturu (ADS) **obousměrně necyklicky 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,

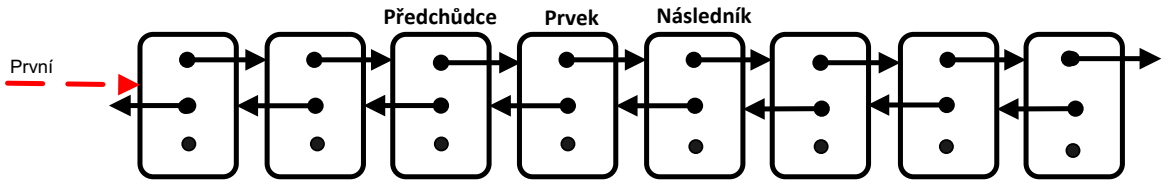
`void vlozPrvni(T data)` – vložení prvku do seznamu na první místo
`void vlozPosledni(T data)` – vložení prvku do seznamu na poslední místo,
`void vlozNaslednika(T data)` – vložení prvku do seznamu jakožto následníka aktuálního prvku,
`void vlozPredchudce(T data)` – vložení prvku do seznamu jakožto předchůdce 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 zpřístupni, přenastavují pozici aktuálního prvku

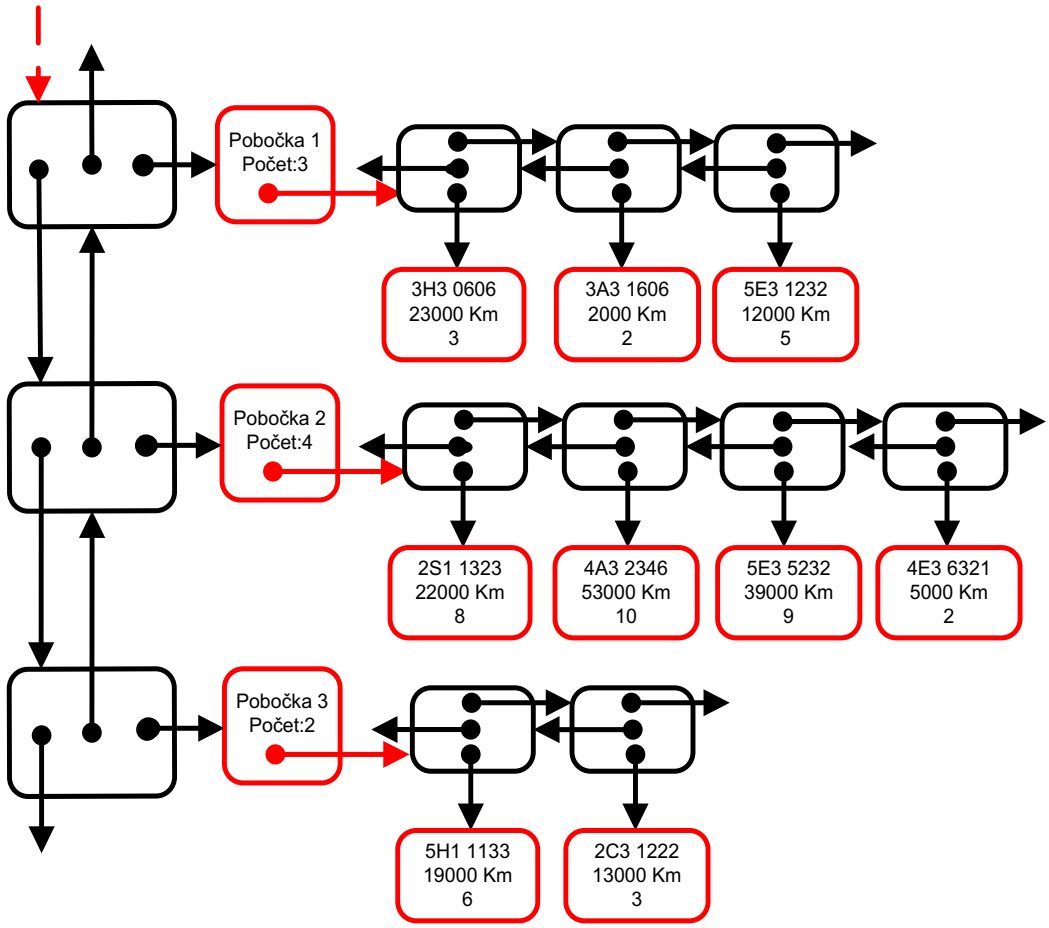
`T odeberAktualni()` – odebrání (vyjmutí) aktuálního prvku ze seznamu poté je aktuální prvek nastaven na první prvek
`T odeberPrvni()` – odebrání prvního prvku ze seznamu,
`T odeberPosledni()` – odebrání posledního prvku ze seznamu,
`T odeberNaslednika()` – odebrání následníka aktuálního prvku ze seznamu,
`T odeberPredchudce()` – odebrání předchůdce aktuálního prvku ze seznamu,

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



Obrázek 1: Obousměrně necyklicky zřetězený lineární seznam

C) Vlastní implementace obsahuje seznam seznamů. Primární seznam reprezentuje jednotlivé autopůjčovny, které obsahují sekundární seznamy automobilů. Stylisticky znázorněno na obrázku 2



Obrázek 2: Seznam seznamů

D) Základním datovým prvkem je datová entita, která je zastoupena abstraktní třídou `Auto`. Auta mohou být **osobní** s atributy (i) SPZ, (ii) stav Km, (iii) počet vypůjčení (iv) barva, nebo **užitková** s atributy (i) SPZ, (ii) stav Km, (iii) počet vypůjčení, (iv) nosnost

E) Dalším datovým prvkem je pobočka, která je reprezentována třídou **`Pobocka`** obsahující (i) jméno pobočky a (ii) počet aut v pobočce (iii) sezname aut na pobočce. Třída `Pobocka` implementuje rozhraní `IPobocka`

`void vlozAuto(Auto auto, EnumPozice Pozice)` – vloží nové auto do seznamu na příslušnou pozici (první, poslední, předchůdce, následník)

`Auto zpristupnAuto(EnumPozice Pozice)` – zpřístupní auto z požadované pozice (první, poslední, předchůdce, následník, aktuální),

`Auto odeberAuto (EnumPozice Pozice)` – odebere auto z požadované pozice (první, poslední, předchůdce, následník, aktuální),

`Iterator iterator()` –vytvoří iterátor

`void zrus()` – zruší všechny auta.

F) Seznam jednotlivých poboček (seznam seznamů) a seznam vypůjčených aut uchovává třída **`Autopujcovna`** implementující rozhraní **`Autopujcovna`**

`void vlozPobocku(IPobocka Pobocka, EnumPozice Pozice)` – vloží novou pobočku do seznamu na příslušnou pozici (první, poslední, předchůdce, následník)

`IPobocka zpristupniPobocku(EnumPozice Pozice)` – zpřístupní pobočku z požadované pozice (první, poslední, předchůdce, následník, aktuální),

`IPobocka odeberPobocku (EnumPozice Pozice)` – odebere pobočku z požadované pozice (první, poslední, předchůdce, následník, aktuální),

`void vlozAuto(Auto auto, EnumPozice Pozice)` – vloží nové auto do seznamu aktuální pobočky na příslušnou pozici (první, poslední, předchůdce, následník)

`Auto zpristupnAuto(EnumPozice Pozice)` – zpřístupní auto z požadované pozice aktuální pobočky (první, poslední, předchůdce, následník, aktuální),

`Auto odeberAuto (EnumPozice Pozice)` – odebere auto z požadované pozice (první, poslední, předchůdce, následník, aktuální),

`Auto vypujcAuto (EnumPozice Pozice)` – odebere auto z požadované pozice aktuální pobočky a vloží ho do seznamu výpůjček (první, poslední, předchůdce, následník, aktuální),

`Auto vratAuto (EnumPozice Pozice)` – odebere auto z požadované pozice výpůjček a vloží ho do seznamu aktuální pobočky (první, poslední, předchůdce, následník, aktuální),

`Auto zpristupniVypujceneAuto(EnumPozice Pozice)` – zpřístupní auto z požadované pozice ze seznamu vypůjčených aut (první, poslední, předchůdce, následník, aktuální),

`Iterator iterator(eTyp typ)`– vrací požadovaný iterátor Poboček/Automobilů/Vypůjčených automobilů

`void zrusPobocku()` – zruší všechny auta v aktuální pobočce

`void zrus()` – zruší všechny pobočky.

Výčtový typ `EnumPozice` je dán stavy: první, poslední, předchůdce, následník, aktuální)

G) Pro obsluhu aplikace vytvořte uživatelské **GUI** rozhraní **ProgAutopujcovna**, které umožňuje obsluhu programu a volat požadované operace.

Zmíněný program nechť umožňuje zadávání vstupních dat z klávesnice, ze souboru (pouze stavy instancí datových objektů) a z generátoru, výstupy z programu nechť je možné zobrazit na obrazovce a uložit do souboru (pouze stavy instancí datových objektů).