**SEMESTRÁLNÍ PRÁCE A**
**PRO SKUPINY K. ŠIMERDY**

Maximální možný bodový zisk: **4 body**

A) Motivační příklad:
Požaduje se pro terapeutickou kliniku vytvořit „desktopovou“ aplikaci, která bude spravovat časový plán termínů terapií pro jednoho terapeuta. Stejné terapie se můžou opakovat v požadovaném období, například dlouhého 1-3 týdnů, přičemž se mohou míchat v daném období i s jinými typy terapií. Délka každé terapie bude buď 2h nebo 4h. Terapie v celkové délce trvání se musí vejít do pracovní doby terapeuta, například 8-16h. Jednotlivé termíny se budou řadit v seznamu termínů terapií podle data a času začátku termínů. Dále bude možné požádat v požadovaném období o zobrazení obsazených a volných hodinových časových oken terapeuta.

B) Použité datové struktury:
Implementujte třídu `AbstrDoubleList` jako abstraktní datovou strukturu (ADS) **obousměrně necyklicky zřetězený lineární seznam** (stylizovaně znázorněný v rámci obr. 1). Tato třída implementuje rozhraní `DoubleList`, které dědí rozhraní `Iterable`. Rozhraní `DoubleList` s typovým parametremT je definováno následovně:

`int getMohutnost()` - vrací počet vložených dat do seznamu,
`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,

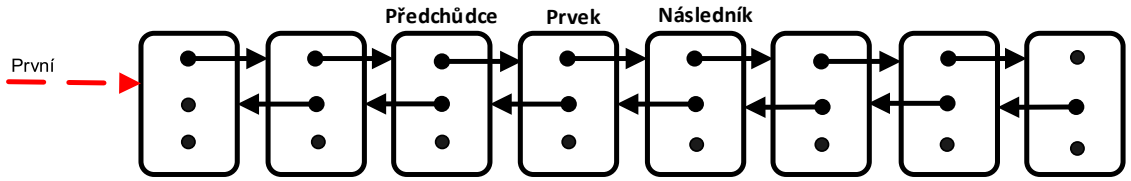
`T zpristupniAktualni()` – zpřístupnění aktuálního prvku seznamu,
`T zpristupniPrvni()` – zpřístupnění prvního prvku seznamu,
`T zpristupniPosledni()` – zpřístupnění posledního prvku seznamu,
`T zpristupniNaslednika()` – zpřístupnění následníka aktuálního prvku,
`T zpristupniPredchudce()` – zpřístupnění předchůdce aktuálního prvku,
Poznámka: Operace typu zpřístupni, mění pozici aktuálního prvku. Výjimkou jsou odebírací metody, když současně mohou ukazovat na aktuální prvek seznamu .

`T odeberAktualni()` – odebrání (vyjmutí) aktuálního prvku ze seznamu, následně 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`)

Pro ověření implementační třídy `AbstrDoubleList` se požaduje, aby třída implementovala rozhraní v souboru `DoubleList.java`, které bude předáno na prvním

cvičení. Rozhraní v `DoubleList.java` může obsahovat další metody nebo jiné signatury, než je zde uvedeno. Pro implementaci platí to, co je uvedeno v kontraktu v souboru s rozhraním.




*Obr. 1: Obousměrně necyklicky zřetězený lineární seznam*

C) Pro ověření funkčnosti implementované ADS vytvořte třídu `SpravaTerminu`**.** Tato třída bude umožňovat správu termínů konkrétního terapeuta, kdy uchovává jeho jméno a příjmení a implementuje následující rozhraní `Sprava,` které dědí rozhraní `Iterable`:

`void vlozTermin(Termin termin, enumPozice pozice)` – prvně metoda ověří zda je možné termín vložit a to pomocí metody `jeVolno`. Pokud je termín volný, pak ho vloží na požadovanou pozici (první, poslední, předchůdce, následník).

`void vlozTermin(Termin termin)` – prvně metoda ověří, zda je možné termín vložit a to pomocí metody `jeVolno`. Pokud je termín volný, pak ho vloží do seznamu na příslušnou pozici dle data začátku termínů. Vždy jsou logovány všechny operace, kterými je toho dosáhnuto.

`Termin zpristupniTermin(enumPozice pozice)` – zpřístupní termín z požadované pozice (první, poslední, předchůdce, následník, aktuální)

`Termin odeberTermin (enumPozice pozice)` – odebere termín z požadované pozice (první, poslední, předchůdce, následník, aktuální)

`boolean jeVolno (LocalDateTime odKdy, LocalDateTime doKdy)` – ověřuje volnost termínu

`Iterator<Termin> iterator()` – vrátí iterátor na procházení seznamu termínů

`MaticeObsazenosti getObsazenost(LocalDate odKdy, LocalDate doKdy)` – tato metoda sestaví matici časové obsazenosti jednotlivých dnů, Osa Y reprezentuje datum od-do dle vstupního parametru, osa X pak čas v rozmezí 8-16h. Matice bude v GUI vhodně vizualizována.

`Termin najdiDalsiVolnyTermin(LocalDate odkdy, LocalDate dokdy)` – nalezne první volný termín, který je ohraničeny vstupním intervalem např. jedno měsíce a to v rozsahu 1-3 týdny o délce 2 nebo 4 hodiny. Termín nesmí mít časový konflikt s již exitující, termín.

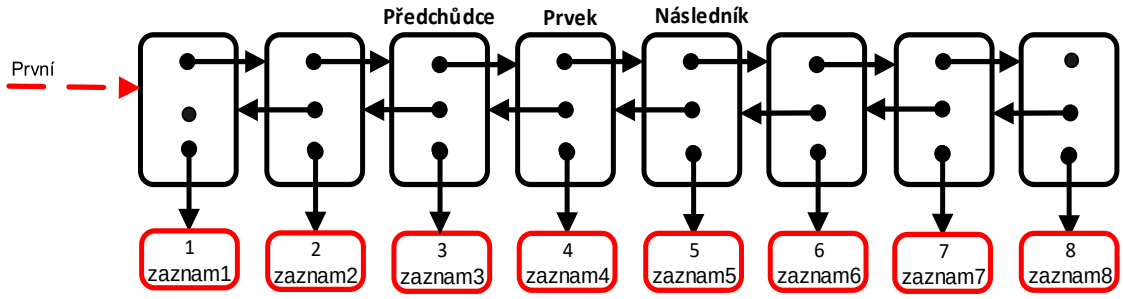
`void generuj(Obdobi obdobi)` – vkládá náhodně generované termíny v zadaném období. Třída období obsahuje počáteční a koncové datum.

`void uloz(String soubor)` – uloží termíny do souboru

`void nacti(String soubor)` – načte uložené termíny do prázdného seznamu

`void zrus()` – zruší všechny termíny.

Pro ověření implementační třídy `SpravaTerminu` se požaduje, aby třída implementovala rozhraní v souboru `Sprava.java`, které bude předáno na prvním cvičení. Rozhraní v `Sprava.java` může obsahovat další metody nebo jiné signatury, než je zde uvedeno. Pro implementaci platí to, co je uvedeno v kontraktu v souboru s rozhraním.




*Obr. 2: Stylizované znázornění seznamu s daty*

Třída `Termin` – bude obsahovat tyto atributy:

```
Terapie terapie;
TrvaniTerapie trvani;
final LocalDateTime start;
final LocalDateTime end;
```

Kde

```
public enum Terapie {
    HYPOTERAPIE("Hypoterapie"),
    CANISTERAPIE("Canisterapie"),
    ARTETERAPIE("Arteterapie"),
    MUZIKOTERAPIE("Muzikoterapie"),
    AQUATERAPIE("Aquaterapie"),
    REFLEXTERAPIE("Reflexoterapie");
…
}

public enum TrvaniTerapie {
    KRATKA("krátka terapie", 2),
    DLOUHA("dlouhá terapie", 4);

    private final String nazev;
    private final int trvani;
  …
}
```

Třída `Termin` bude navržena tak, aby její instance byly hodnotové neměnné objekty.

D) Vytvořte grafické uživatelské rozhraní v JavaFX, které bude umožňovat

1. ovládání metod deklarovaných v rozhraní `Sprava`,
2. zobrazení termínů a matice obsazenosti.

Požaduje se

3. zaznamenávání výjimek a událostí do logovacího souboru
4. implementací tříd rozhraní `DoubleList` a `Sprava` ověřit jednotkovými testy v `JUnit`.