JAVA programavimo kalba

Java kolekcijos (collections) ir final reikšmė

Kolekcijos (collections) duomenų tipai

- Java kolekcijos - tai klasės - konteineriai, kuriuose galima laikyti keletą elementų. Kažkas panašaus kaip masyvas, tik elementų skaičius tame konteineryje gali keistis.
- Paprastai kolekcijose laikomi vieno tipo arba vieno bazinio tipo objektai.
- Kolekcijose negalima laikyti primityvių tipų duomenų, t.y. int, double, boolean ar char. Bet tam galima panaudoti atitinkamas pakuočių klases - Integer, Double, Boolean ar Character.

Kolekcijų karkasas (collections framework)

- Java kolekcijų karkasas (biblioteka) - tai rinkinys sąsajų (interfaces), klasių ir statinių funkcijų.
- Sąsajos (interfaces) - tai abstraktūs duomenų tipai vaizduojantys įvairias kolekcijas (sąrašas, eilė, stekas, žodynas, …)
- Kolekcijų klasės - tai kolekcijų sąsajų įvairios realizacijos (implementations).
- Kolekcijų funkcijos - realizuojančios įvairius universalius algoritmus, kaip kad rūšiavimo, paieškos ir pan.

Kolekcijų tipai (collections types)

- List
- Set
- Map
- Queue
- Deque

Kolekcijos

- Kai aprašome kolekciją, tai reikia nurodyti jos tipą ir taip pat kokio tipo (klasės) bus jos elementai (o taip pat ir kokia raktų klasė jei tai Map kolekcija).
- Tai nurodoma prie kolekcijos tipo rašant: < *elemento-klasė* > arba < *rakto-klasė, elemento-klasė* >
- List<Integer> a;
- Map<String, Zmogus> zmones;
- Kai kolekciją kuriam, tai tipus nurodyti nėra privaloma, nes java kompiliatorius pats supranta kokie tipai turi būti pagal tai koks kintamojo arba parametro tipas.
- Todėl užtenka nurodyti tik < >
- List<Integer> a = new ArrayList<>();
- Map<String, Zmogus> zmones = new HashMap<>();

List - collection

- List - tai sąrašą ir darbą su juo aprašanti sąsaja.
- Sąrašas - tai elementų kolekcija, turinti šias savybes:
  - Elementai sąraše turi tam tikras vietas (panašiai kaip masyve), kurios nesikeičia be mūsų įsikišimo.
  - Elementus sąraše galima pasiekti pagal jų eilės numerį
  - Elementus sąraše galima pasiekti einant nuo sąrašo pradžios iki galo
  - Galima patikrinti ar sąraše nėra tam tikro elemento
- Sąrašo sąsają realizuoja daug klasių: Vector, ArrayList, …

Set - collection

- Set - tai aibę (arba nesikartojantį sąrašą) ir darbą su juo aprašanti sąsaja
- Pagrindinis skirtumas nuo sąrašo yra tas, kad mes nežinome koks elemento eilės numeris ar indeksas, tiesiog tokios savybės aibėje nėra, nes ji saugo elementus kitaip.
- Aibės yra dviejų rūšių:
  - Paprasta - kada mes nežinome ir mums nerūpi elementų išsidėstymo tvarka
  - Lygiuota - kai elementai išrikiuojami sąraše pagal jų palyginimus (compareTo metodas)
- Paprastą aibės sąsają realizuoja HashSet, o lygiuotą TreeSet. Yra ir daugiau realizacijų.

Map - collection

- Map - tai sąsaja, kuri aprašo kaip turi veikti kolekciją tipo “raktas , reikšmė”
- Kartais toks kolekcijos tipas vadinamas “žodynu” (dictionary).
- Javoje yra ir kolekcijos sąsaja Dictionary, bet jos realizacijos pasenusios ir rekomenduojamos nenaudoti. Vietoj jų naudoti rekomenduojama Map.
- Kaip ir aibės yra paprastas ir lygiuotas variantai raktų (!) atžvilgiu
- Paprastą Map sąsają realizuoja HashMap, o lygiuotą TreeMap.
- Yra ir daugiau realizacijų.

Queue/Deque collections

- Queue, Deque - tai sąsajos panašios į sąrašą tik turinčios papildomas funkcijas ir apribojimus kaip elementai gali būti pridėti ir išimti.
- Queue - eilę aprašanti sąsaja dirbanti dažniausia FIFO (firstin-firstOut) metodu
- Deque yra labai panaši kolekcija į Queue tik dirba iš abiejų eilės galų
- Queue sąsają realizuoja LinkedList, o Deque - ArrayDeque

Final reikšmė kintamiesiems (final variables)

Jeigu norim pabrėžti, kad klasės lauko, kintamojo arba metodo parametro reikšmės negalima pakeisti, tai reikia prie kintamojo aprašo arba metodo parametro nurodyti modifikatorių final

```
void method(final int a) {
        a += 2;
}
final double pi = 3.1415926536;;
pi = 3.0;
```

Final reikšmė metodams (final methods)

Paprastai dukterinėje klasėje (subklasėje) galima iš naujo aprašyti (realizuoti) metodą, kuris egzistuoja tėvynėje klasėje (superklasėje).

Bet jei mes esame tos superklasės autorius ir nenorime, kad mūsų metodas būtų realizuotas kitaip bet kurioje subklasėje, tai mes tokį metodą galima pažymėti modifikatoriumi final ir jo nebus galima pakeisti.

```
class A {
        void int methodA(…) {…}
        final void methodB(…) { … }
}


class B extends A {
        void int methodA(…) {…}
        void methodB(…) { … }
}
```

Final reikšmė klasei (final class)

Būna taip, kad mes aplamai nenorime ne tik kad metodas būtų perrašytas, bet ir visa klasė praplėsta, t.y. norime uždrausti iš mūsų klasės kurti dukterines klases. Tokiu atveju klasę reikia pažymėti modifikatoriumi final

final class A {}

class B extends A {}

Konstantos (constants)

Jei mes norime apsirašyti konstantas, t.y. tokius kintamuosius, kurių reikšmės niekada nesikeičia ir prie jų galima prieiti nekuriant jokių abjektų, tai geriausia tai padaryti kombinuojant modifikatorius static ir final

```
class Const {
        static final double PI = 3.1415926536;
        static final String VERSION = “1.2”;
}
double area = r * r * Const.PI;
```

Uždaviniai

1. Sukurkite žodyno tipo kolekciją saugoti žmonių klasės objektus (su tokiais laukais: vardas, pavardė, asmens kodas), o kaip raktą naudokite asmens kodą.
2. Įdėkite keletą žmonių į kolekciją ir atspausdinkite žmones asmens kodo didėjimo tvarka.
3. Pabandykite įdėti į kolekciją du skirtingus žmones bet su tuo pačiu asmens kodu. Patikrinkite kas atsitiks?
4. Pagalvokite kaip saugoti žmones jei norime turėti kelis su tuo pačiu asmens kodu.
5. Sukurkite kavines aptarnavimo programa. Pvz: Kavine turi staliukus ir patiekalus. Ateina klientas ir atsiseda prie staliuko. Ji aptarnauja padaveja. Klientas uzsisako patiekalus / gerimus. Padaveja isijungia aplikacija ir suveda duomenis. Kiekvienas patiekalas / gerimas turi kaina. Klientas uzsako saskaita. Padaveja apskaiciuoja jam ir pateikia saskaita.

   Atspausdinkite :

   Staliuko Nr.

   Uzsakytu patiekalu skaicius ir ju pavadinimas

   Uzsakytu gerimu skaicius ir ju pavadinimas

   Saskaitos suma, kuria turi sumoketi klientas

   Arbatpinigiu suma

   1. Pvz: Yra 10 staliuku su numeriais {5, 6, 2, ….} .
   2. Apsilanko N klientu. Klientu skaicius gali keistis. Tai galite padaryti ivesdami konsoleje reiksme arba priskirdami kintamajam kiekviena karta paleidus programa vis skirtinga sugalvota reiksme.
   3. Klientas uzsisako N patiekalu ir N gerimu.
   4. Kitas Klientas taip pat gali uzsisakyti N patiekalu ir N gerimu ir t.t (patikrinti ar veikia , kiekvienam klientui suteikite skirtinga kieki patiekalu ir gerimu)
   5. Naudokite viska ka iki siol ismokote ;)