JAVA programavimo kalba

Java sąsaja (interface) ir klasės pakuotės (wrapper classes)

Sąsaja (interface)

- Vienas būdas realizuoti abstrakcijas yra su abstrakčiomis klasėmis
- Kitas būdas - tai interfeisai

```
public interface Area {
public void setSizeFromArea(double area);
}
```

- Pagal nutylėjimą interfeise nurodyti metodai turi public prieinamumą
- Interfeiso aprašas primena abstrakčios klasės aprašymą, kur visi metodai yra abstraktūs.

Sąsaja (interface)

- Kaip naudoti interfeisus aprašant klases?

Aprašant mes turime nurodyti, kad mūsų klasė įgyvendina interfeisą, t.y. jame aprašytus metodus

class Circle extends Figure **implements** Area {

}

- Mes privalome klasėje Circle pilnai aprašyti visus metodus nurodytus sąsajoje Area. Prie jų taip pat labai naudinga nurodyti žymę (anotaciją) @Override

Sąsajos praplėtimas (interface extends)

Kaip ir klasės taip ir sąsajos gali išplėsti/papildyti kitas sąsajas papildomais metodais

interface Geometry **extends** Area {

void setSizeFromPerimeter(double perimeter);

…

}

Kelių sąsajų įgyvendinimas (class implements multiple interfaces)

Kai žinote, klasė gali būti dukterine klase tik vienai klasei, t.y. ji gali turėti vieną ir tik vieną tėvinę klasę.

- Bet klasė gali įgyvendinti ne vieną, o kelias sąsajas

class Figure **implements** Area, Color, Border {

…

}

- Toks užrašas reiškia, kad klasėje Figure aprašyti/įgyvendinti visi metodai nurodyti ir Area ir Color ir Border sąsajose.

Sąsajos default metodai (interface default methods)

- Sąsajoje galima turėti ne tik metodų aprašus (signature), bet ir jų įgyvendinimą (implementation).
- Tokie metodai sąsajoje pažymimi modifikatoriumi default
- Tokiu būdu klasėje, kuri įgyvendina sąsają, nebūtina (!) tokį metodą įgyvendinti.

Sąsajos default metodai (interface default methods)

- Jei klasė įgyvendina dvi sąsajas, kurios abi turi tokį patį default metodą (t.y. toks pats pavadinimas ir parametrai), tai klasė privalo įgyvendinti metodą, nes java nežino kurį iš metodų naudoti.

Sąsajos static metodai (interface static methods)

- Panašiai kaip sąsajos default metodus, galima aprašyti static metodus
- Nuo default jie skiriasi tuo, kad pasidaro priskirti ne prie objekto egzemplioriaus (instance), o prie klasės - kaip kad klasės metodai
- Static metodai gali būti kviečiami iš kitų sąsajos metodų, arba kaip sąsajos metodai (kaip kad klasės metodai).

Sąsajos private metodai (interface private methods)

Java9 jau galima apibrėžti privačius (private) metodus, kurie veikia kaip ir default bet yra prieinami iš kitų sąsajos metodų, bet ne iš klasės ar klasės objekto.

- Analogiškai kaip static metodai galima aprašyti private static metodus, kurie taip pat bus prieinami tik iš sąsajos.

Sąsajos tipo kintamieji / parametrai (interface as a type or as a parameter)

- Sąsają galima naudoti nurodant kintamųjų ar metodo parametrų tipus
- Tokio tipo kintamiesiems ar parametrams galima priskirti tokius objektus, kurių klasė įgyvendina duotą sąsają (t.y. kurių klasė implementuoja duotą interfeisą).

Skaičiai – klasės pakuotės (wrapper classes)

Dirbant su skaičiais, dažniausiai tenka naudoti primityvius duomenų tipus, kaip kad int, long, double ir pan.

Tačiau kartais naudinga vietoj jų turėti taip vadinamas klases pakuotes, t.y. tokias klases, kurios kaip ir “supakuoja” atitinkamus primityvius tipus:

- byte - Byte
- short - Short
- int - Integer
- long - Long
- float - Float
- double – Double
- boolean - Boolean
- char - Character

Klasės pakuotės (wrapper classes)

- Kam čia tokios klasės reikalingos? pvz. duomenų bazės laukelis, kuriame saugojamos kažkokios skaitinės reikšmės neturi jokios reikšmės ir todėl su primityviais tipais neįmanoma išsiversti.
- Iš programuotojo perspektyvos žvelgiant, pakuotės klasės skiriasi nuo primityvių tipų tik tuo, kad atitinkamos klasės kintamasis gali aplamai neturėti jokios reikšmės (null).

Skaičių pakuotė (Number wrapper class)

- Skaičių klasės turi vieną bendrą tėvinę klasę Number. Klasė Number turi ir daugiau vaikų:
- AtomicInteger, AtomicLong - skirtos dirbti iš daugelio gijų (multithread)
- BigInteger, BigDecimal - skirtos atlikti veiksmus su labia dideliais tikslumais.

BigDecimal tipas

BigDecimal labai naudinga klasė jei reikia atlikti skaičiavimus norimu tikslumu - pvz. skaičiuoti pinigus. Kai kuriamas BigDecimal galima nurodyti sveiką skaičių (**unscaledVal** - int, Integer arba BigInteger tipo) ir tikslumą (**scale**):

- BigDecimal a1 = BigDecimal.valueOf(1234, 2);

arba skaičių galima nurodyti kaip tekstinę eilutę:

- BigDecimal a2 = new BigDecimal(“12.34”);

BigDecimal tipas

- Nerekomenduojame naudoti BigDecimal konstruktorių su double tipo argumentu dėl to, kad kompiuteryje tokie skaičiai visada apvalinami ir saugojami ne tiksliai tokie kokius mes juos įvedame.
- Jei sukonstruoti BigDecimal skaičių su kableliu tai geriau naudoti konstruktorių su tekstinės eilutės argumentu arba konstruktorių su sveikais skaičiais - reikšme ir skale.

Autoboxing / unboxing

- Ten kur reikia, java kompiliatorius automatiškai bando versti primityvų tipą į atitinkamos klasės objektą (autoboxing) ir atvirkščiai (unboxing).

Naudingi pakuočių klasių metodai

Klasės pakuotės turi nemažai naudingų metodų ar konstančių, pvz.:

- MIN_VALUE, MAX_VALUE - atitinkamos klasės galimai mažiausia ir didžiausia reikšmės.
- compare(x, y) ir compareTo(y) - metodai palyginantys vieną reikšmę su kita ir grąžinantys int atitinkamai < 0, 0 ir > 0, jei pirmasis skaičius yra mažesnis, lygus ar didenis už kitą skaičių.
- valueOf(…) - sukonstruojamas atitinkamos klasės objektas iš kitų panašių duomenų ar tekstinės eilutės - analogas konstruktoriui.
- parseInt(String s), parseDouble(String s),… - atitinkami metodai iš tekstinės eilutės konstruoti primityvius tipus.

Uždaviniai

1. Aprašykite interfeisą Mokėjimas (Payment) su šiais metodais:
   - banko sąskaita (bank account) - grąžina banko sąskaitos numerį.
   - sąskaitos turėtojas (account owner) - fizinio ar juridinio asmens pavadinimas.
   - suma (amount) - pervedama suma.
2. Sukurkite klases Darbuotojas (Employee) ir Klientas (Client), kurios implementuoja Mokėjimas interfeisą. Sukurkite masyvą iš keleto darbuotojų ir klientų objektų ir cikle išveskite jų mokėjimo informaciją.
3. Tarkime yra parduota kažkiek prekių (x) už kažkokią kainą (y) ir tai sudaro kažkokią sumą (z). Visos šitos sumos yra su PVM 21%. Parašykite metodą/funkciją kurią iškvietus su parametrais: kiekis, suma su PVM - ta funkcija atspausdintų pardavimo įrašą tokiu formatu:
   - prekės kaina be PVM (tikslumas 2 ženklai po kablelio)
   - kiekis
   - suma be PVM (tikslumas 2 ženklai po kablelio)
   - PVM suma (tikslumas 2 ženklai po kablelio)
   - viso suma su PVM (tikslumas 2 ženklai po kablelio).