Harjutustund 11

Abstraksed klassid.

Abstraktne klass on klass, mis sisaldab vähemalt ühte abstraktset meetodit. Lisaks võib klass sisaldada ka mitteabstraktseid meetodeid.
Abstraktne meetod on meetod, millel ei ole realisatsiooni (sisu). Sellise meetodi loomiseks kasutatakse märksõna ***abstract***. Abstraktsel meetodil on järgmine üldkuju:

```
abstract piiritlejad tagastustüüp meetodiNimi(parameetrite loetelu);
```

Abstraktse klassi alusel ei saa luua objekte nagu tavalise klassi põhjal. Objektide loomiseks tuleb abstraktsed meetodid realiseerida mitteabstraktse klassi poolt. See tähendab, et tuleb luua abstraktset klassi laiendav alamklass (kasutatakse nagu tavalise pärimise puhul märksõna ***extends***) ning abstraktsetele meetoditele selles sisu kirjutada.

Ülesanne

Tarkvaraarenduse üksuse juht loob uut süsteemi oma töötajate haldamiseks. Varem tal seda süsteemi vaja ei läinud, sest reeglid olid kõik tema peas. Kuid nüüd, kus firma areneb väga kiiresti, töötajate arv kasvab ja otsuseid teeb suurem inimeste ring, on vaja need reeglid infosüsteemiks koondada. See süsteem tegeleb peaasjalikult arendajate palga- ja töötingimuste arvutamisega ja aitab otsustada arendajate palgataset ning boonuseid. Valemid on välja töötatud aastatepikkuse kogemuse põhjal.

1. Looge testklass, kus sees on main meetod.

2. Looge paketti developers abstraktne klass Developer koos konstruktoriga, mis võtab argumendiks arendaja nime, töökogemuse täisaastates ja aine „Objektorienteeritud programmeerimine” hinde. Salvestage need väärtused privaatsetele klassi väljadele.

3. Looge kõigi kolme klassivälja jaoks avalikud getter-meetodid.

4. Looge klassi Developer järgmised package-private nähtavusega abstraktsed meetodid:
   - meetod, mis tagastab vaikimisi pakutava palga
   - meetod, mis tagastab, kas arendaja on suuteline osalema kriitilise tähtsusega projektides

5. Kirjutage Developer klassi package-private nähtavusega meetod getGradeCoefficient(), mis tagastab vastavalt OOP aine hindele koefitsendi:
   - Kui hinne on 1 või 2, tagastab 0
   - kui hinne on 3, tagastab 1
   - kui hinne on 4, tagastab 2
   - kui hinne on 5, tagastab 4

6. Looge klassid JuniorDeveloper ja SeniorDeveloper, mis laiendavad Developeri.
   NB! Klassid ei päri üksteiselt konstruktoreid! Kui peaklassil on argumentidega konstruktor, peate alamklassis konstruktori looma ja peaklassi konstruktori ise välja kutsuma.

7. Realiseerige mõlemas klassis meetod, mis tagastab vaikimisi pakutava palga järgmise valemi alusel:
   JuniorDeveloper: 800 + getGradeCoefficient * 75
   SeniorDevloper: 1200 + getGradeCoefficient * 100 + tööstaaž * 50

8. Realiseerige mõlemas klassis meetod, mis tagastab, kas arendaja on suuteline osalema kriitilise tähtsusega projektides järgmiste reeglite alusel:
   - kui OOP hinne 5, siis on võimeline
   - muul juhul on võimeline ainult siis, kui tööstaaži ja OOP hinde summa on vähemalt 5 ülejäänud juhtudel ei ole võimeline

   Kas need kolm reeglit annab lihtsamaks kokku koondada?

9. Uurige, kuidas mõjutab OOP-aine hinne programmeerija palgataset. Looge testklassis mitu erinevat JuniorDeveloper ja SeniorDeveloper objekti ning trükkige konsooli nende vaikimisi palgatase. NB! Loodud objektide runtime-tüübiks määrake Developer.
   Kas teie loodud süsteemis esineb polümorfismi?

10. Looge klassi JuniorDeveloper meetod, mis tagastab arendaja ennustatava edutamisaja aastates järgmisele palgatasemele:
    - kui OOP hinne 1 või 2 – 4 aastat
    - kui OOP hinne 3 – 3 aastat
    - kui OOP hinne 4 – 2 aastat
    - kui OOP hinne 5 – 1 aasta

    Kutsuge see meetod JuniorDeveloper objektide peal välja. Selle meetodi väljakutsumiseks tuleb esmalt testklassis JuniorDeveloper objektidega veel midagi teha.