Tehtävä 1: Metodit, listat, alkuluvut (4p)

Tässä tehtävässä käsittelet metodeja, listoja sekä alkulukuja (englanniksi *prime*).

Alkuluvut ovat lukuja, jotka ovat suurempia kuin yksi ja jotka ovat jaollisia vain luvulla yksi sekä itsellään. Alkulukuja ovat esimerkiksi 2, 3, 5, 7, 11, 13, 17 jne. Tee tehtäväpohjaan metodi:

```
public static List<Integer> alkulukujenIndeksit(List<Integer> luvut)
```

Metodi `alkulukujenIndeksit` saa parametrinaan listan kokonaislukuja. Metodin tulee palauttaa lista, joka sisältää parametrina saadun listan alkulukujen indeksit.

Alla esimerkki metodin käytöstä.

```
List<Integer> lista = new ArrayList<>();
lista.add(8);
lista.add(4);
lista.add(1);
lista.add(2);
lista.add(5);
lista.add(12);

List<Integer> indeksit = alkulukujenIndeksit(lista);
for(int indeksi :indeksit) {
   System.out.println("Indeksissä " + indeksi +
              " on alkuluku " + lista.get(indeksi));
}
```

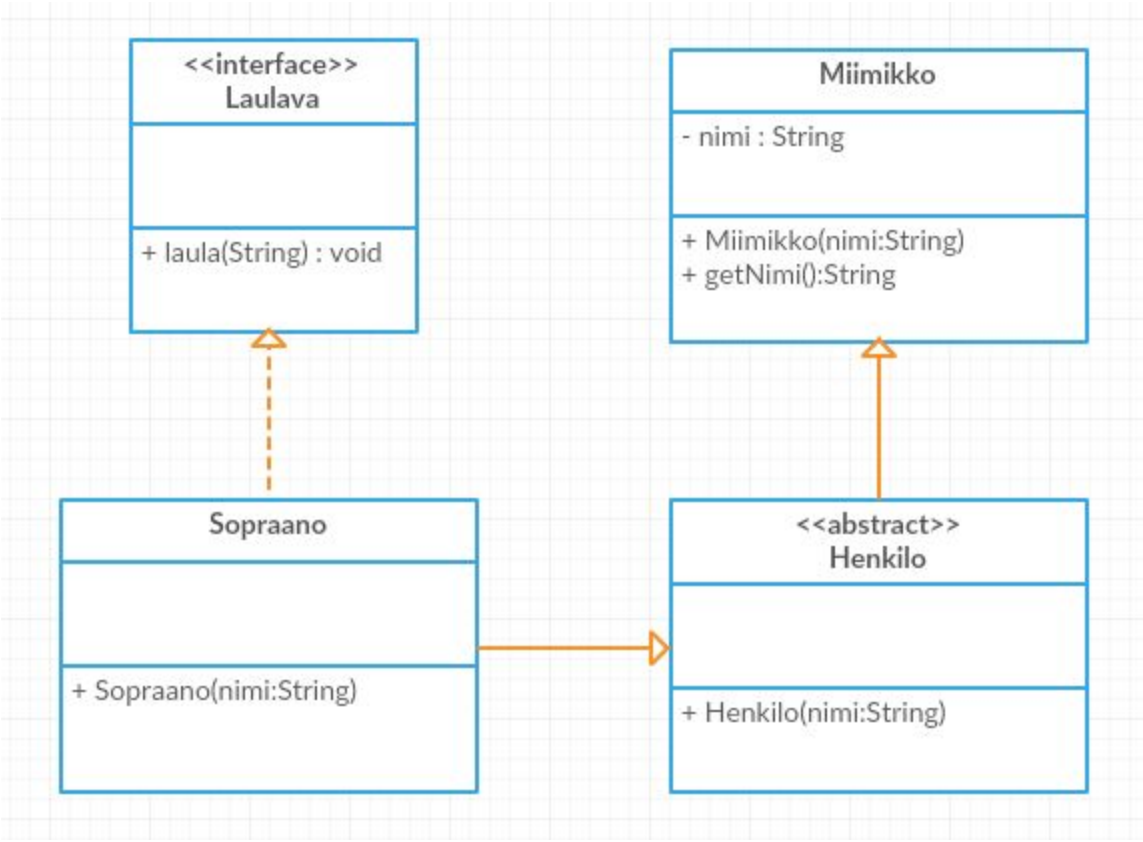
Yllä olevan ohjelman tulostus on seuraava.

```
Indeksissä 3 on alkuluku 2
Indeksissä 4 on alkuluku 5
```

Huom! Tarkista, että metodin nimi on täsmälleen yllä kuvattu.

Tehtävä 2: Luokkakaavion tulkinta (4p)

Alla on kuvattuna luokkakaavio. Toteuta luokkakaavion kuvaama lähdekoodi tehtäväpohjan oletuskansioon (default package).



Tehtävä 3: Jääkiekkotilastot (8p)

Kirjoita ohjelma tiedostossa olevien jääkiekko-otteluiden tuloksiin liittyvien tietojen tulostukseen.

Ohjelma kysyy käyttäjältä avattavan tiedoston nimeä. Tämän jälkeen ohjelma lukee tiedoston ja tulostaa tiedostossa oleviin jääkiekkotuloksiin liittyviä tietoja. Mikäli tiedostoa ei löydy, ohjelma kertoo siitä, ja lopettaa suorituksen -- ohjelman ei tule heittää poikkeusta.

Luettavan tiedoston muoto on seuraava:

```
joukkue1;joukkue2;1;1
joukkue3;joukkue4;2;1
joukkue2;joukkue3;1;3
joukkue4;joukkue1;2;1
```

Kukin tiedoston rivi kertoo yksittäisen ottelun tuloksen. Esimerkin ensimmäinen rivi tarkoittaa, että joukkue1 ja joukkue2 pelasivat tasapelin 1-1. Toinen rivi kertoo että joukkue3 voitti joukkue4:n 2-1. Kolmas rivi kertoo, että joukkue2 hävisi joukkue3:lle 1-3, eli toisin sanoen joukkue3 voitti joukkue2:n 3-1. Rivien määrä ei ole ennalta rajattu.

Luettuaan syötteen, ohjelman tulee tulostaa tiedoston perusteella (1) joukkueiden nimet aakkosjärjestyksessä ja (2) otteluiden tuloksen pohjalta lasketun sarjataulukon.

(1) Joukkueiden nimet tulee tulostaa aakkosjärjestyksessä siten, että kunkin joukkueen nimi esiintyy tasan kerran. Joukkueiden nimien tulostusta edeltävällä rivillä tulee olla merkkijono “Joukkueet”.

(2) Sarjataulukon jokainen rivi tulostetaan siten, että ensin tulee joukkueen nimi, sen jälkeen joukkueen voitot, tappiot, tasapelit ja lopuksi joukkueen pisteet. Pisteet lasketaan siten että voitosta saa kolme pistettä, tappiosta ei yhtään ja tasapelistä yhden. Esim. yllä joukkue3 on voittanut kaksi peliä, mutta ei ole hävinnyt yhtään tai pelannut yhtäkään tasan. Täten joukkue3:n rivillä tulee olla luvut 2, 0, 0 ja 6. Sarjataulukossa joukkueiden tiedot tulee tulostaa pistejärjestyksessä. Mikäli useammalla joukkueella on sama pistemäärä, nämä joukkueet tulostetaan aakkosjärjestyksessä. Sarjataulukon tulostusta edeltävällä rivillä tulee olla merkkijono “Sarjataulukko”.

Esimerkkitulostus yllä kuvatulla tiedostolla (tässä tiedot löytyvät tiedostosta tulokset.txt).

```
Syötä luettava tiedosto: tulokset.txt

Joukkueet
joukkue1
joukkue2
```

```
joukkue3
joukkue4

Sarjataulukko
joukkue3 2 0 0 6
joukkue4 1 1 0 3
joukkue1 0 1 1 1
joukkue2 0 1 1 1
```

Alla vielä toinen esimerkkitiedosto ja esimerkkitulostus. Tässä oletetaan, että tiedot löytyvät tiedostosta nimeltä kisat.txt).

```
suomi;etelä-korea;8;1
norja;latvia;2;2
suomi;latvia;8;1
```

Esimerkkitulostus

```
Syötä luettava tiedosto: kisat.txt

Joukkueet
etelä-korea
latvia
norja
suomi

Sarjataulukko
suomi 2 0 0 6
latvia 0 1 1 1
norja 0 0 1 1
etelä-korea 0 1 0 0
```

Vihje: saat pilkottua tiedostossa olevat merkkijonot seuraavasti:

```
String rivi = “suomi;etelä-korea;8;1”
String[] palat = rivi.split(";");
// palat[0] on nyt merkkijono suomi
// palat[1] on nyt merkkijono etelä-korea
// palat[2] on nyt merkkijono 8
// palat[3] on nyt merkkijono 1
```

Huom! Joudut luomaan testaukseen mahdollisesti käyttämäsi tiedostot itse.

Tehtävä 4: Lippuvaraamo (8p)

Kirjoita ohjelma matkalippujen varaamiseen.

Ohjelma tarjoaa käyttäjälle mahdollisuuden (1) matkojen lisäämiseen, (2) matkojen listaamiseen, (3) matkan varaamiseen sekä (4) matkakohtaisten varausten tulostamiseen.

(1) Matkaa lisätessä käyttäjältä kysytään lähtöpaikkaa, kohdepaikkaa, sekä istumapaikkojen määrää. Järjestelmän tulee luoda matkan lisäyksen yhteydessä matkalle numerotunnus, jota voi käyttää varauksen yhteydessä.

(2) Matkojen listaus listaa kaikki matkat sekä niihin liittyvien vapaiden istumapaikkojen lukumäärät. Jokainen matka tulostetaan omalla rivillään. Kukin rivi alkaa matkaan liittyvällä numerotunnuksella, jonka järjestelmä on luonut.

(3) Matkan varaus kysyy käyttäjältä matkaan liittyvää numerotunnusta sekä varaajan sähköpostiosoitetta, varaus onnistuu mikäli kyseisellä matkalla on vielä istumapaikkoja.

(4) Matkakohtaisten varausten tulostus taas kysyy käyttäjältä matkaan liittyvää numerotunnusta ja tulostaa tämän jälkeen kaikki sähköpostiosoitteet, joilla kyseinen matka on varattu.

Alla on esimerkki ohjelman toiminnasta. Käyttäjän syötteet on merkitty punaisella.

```
1) lisaa matka
2) listaa matkat
3) varaa lippu matkalle
4) listaa matkan liput
5) lopeta
(muut tulostavat valikon)

Syötä komento: 1
Mistä: Helsinki
Minne: Joensuu
Paikkoja: 2

Syötä komento: 1
Mistä: Helsinki
Minne: Turku
Paikkoja: 1

Syötä komento: 2
0: Helsinki-Joensuu, paikkoja jäljellä 2
```

```
1: Helsinki-Turku, paikkoja jäljellä 1

Syötä komento: 3
Mikä matka: 1
Sähköposti: matkalla@turkuun.net

Syötä komento: 2
0: Helsinki-Joensuu, paikkoja jäljellä 2
1: Helsinki-Turku, paikkoja jäljellä 0

Syötä komento: 4
Minkä matkan liput listataan? 1
matkalla@turkuun.net

Syötä komento: 3
Mikä matka: 1
Sähköposti: matkalla@taytta.net

Syötä komento: 4
Minkä matkan liput listataan? 1
matkalla@turkuun.net

Syötä komento: 4
Minkä matkan liput listataan? 0

Syötä komento: 5
```

Tehtävä 5: Ohjelman testausta (4p)

Tässä tehtävässä luodaan automaattisia testejä annetulle ohjelmalle.

Tehtäväpohjassa tulee mukana kirjasto, joka tarjoaa tekstipohjaisen käyttöliittymän pankkisovellukselle.

Voit testata pankkisovellusta tälläisellä koodilla:

```
public static void main(String[] args) {
  Pankki.suorita(new Scanner(System.in));
}
```

Suorita-metodi käynnistää tekstuaalisen käyttöliittymän, jonka avulla ohjelmaa käytetään. Kun ohjelman käyttö lopetetaan, metodi palauttaa ohjelman suorituksessa tehdyt tulostukset merkkijonona. Samat merkkijonot tulostetaan myös ohjelman suorituksen aikana.

Tehtävänäsi on selvittää miten pankki toimii ja kirjoittaa sille automaattiset testit. Pohdi minkälaisilla testeillä saat testattua ohjelman toimintaa mahdollisimman kattavasti, mutta kuitenkin niin, että testi kertoo tarkkaan mikä ei toiminut odotetusti.

Kirjoita tehtävään toteuttamasi testit tehtäväpohjan mukana annettuun luokkaan PankkiTest. Luokassa on valmiina muutama esimerkkitesti.