Ohjelmoinnin perusteet, kurssikoe 25.2.2014

Vastaa tehtäviin 1, 2, 3, 4 ja 5 **erillisille** konsepteille. Kirjoita jokaiseen konseptiin kurssin nimi, kokeen päivämäärä, nimi ja opiskelijanumero. Vastaukset palautetaan tehtäväkohtaisiin pinoihin. Vaikka jättäisit johonkin tehtävään vastaamatta, tulee vastauspaperi siinäkin tapauksessa palauttaa.

Voit lyhentää komennon `System.out.println` kirjoittamalla `sout` ja käyttää muitakin lyhenteitä, mikäli selität lyhenteen jokaisessa paperissa, jossa sitä käytät. Voit olettaa kaikissa tehtävissä, että ohjelman runko on seuraava:

```
import java.util.Scanner;

public class Ohjelma {
    public static void main(String[] args) {
       Scanner lukija = new Scanner(System.in);
       // int luku = Integer.parseInt( lukija.nextLine() );
       // String rivi = lukija.nextLine();
    }
}
```

1. Käsitteistö (5p)

   Kerro mitä seuraavat käsitteet tarkoittavat ja miten ne liittyvät toisiinsa: luokka, olio, metodi, oliomuuttuja, metodin apumuuttuja. Anna konkreettinen koodiesimerkki, joka havainnollistaa käsitteitä.

   Tehtävän vastauksen pituuden ei tule ylittää kahta sivua.

2. Ohjelmoinnin perusasiat (7p)

   (a) (3p) Tee ohjelma, joka tulostaa toistolausetta (esim. while tai for) käyttäen kaikki 2:lla jaolliset kokonaisluvut aloittaen luvusta 1000 ja päätyen lukuun 2. Tulostuksen tulee tapahtua siten, että jokaiselle riville tulostetetaan 5 lukua, tämän jälkeen tulostus alkaa seuraavalta riviltä, eli ohjelman tulostuksen tulee näyttää seuraavalta:

   ```
   1000 998 996 994 992
   990 988 986 984 982
   980 978 976 974 972
   (paljon rivejä välissä)
   10 8 6 4 2
   ```

   (b) (4p) Tee ohjelma, joka saa syötteekseen opiskelijoiden koepistemääriä kuvaavia kokonaislukuja. Ohjelman toiminta alkaa siten, että se lukee käyttäjältä pistemääriä. Pistemäärien lukeminen loppuu siihen kun käyttäjä syöttää luvun -1.

   Pistemäärän tulee olla luku väliltä 0-30. Jos ohjelmalle syötetään jokin muu pistemäärä, ohjelma jättää sen huomioimatta.

   Luettuaan pistemäärät ohjelma ilmoittaa, mikä (väliltä 0-30 olleista) pistemääristä oli suurin. Pistemääristä ne joiden suuruus on alle 15 vastaavat arvosanaa *hylätty* ja loput arvosanaa *hyväksytty*. Ohjelma ilmoittaa myös hyväksyttyjen ja hylättyjen arvosanojen lukumäärät.

   Esimerkki:

   ```
   Syötä koepistemäärät, -1 lopettaa:
   20
   12
   ```

```
29
15
-1
paras pistemäärä: 29
hyväksyttyjä: 3
hylättyjä: 1
```

Esimerkissä pistemäärä 12 vastaa hylättyä ja pistemäärät 20, 29 ja 15 hyväksyttyjä suorituksia. Eli ohjelma ilmoittaa että hyväksyttyjä on 3 ja hylättyjä 1.

Huomaa, että ohjelman tulee jättää huomioimatta kaikki välin 0-30 ulkopuolella olevat pistemäärät. Esimerkki toiminnasta tilanteessa, jossa syötteiden seassa on huomioimatta jätettäviä pistemääriä:

```
Syötä koepistemäärät, -1 lopettaa:
10
100
20
-4
30
-1
paras pistemäärä: 30
hyväksyttyjä: 2
hylättyjä: 1
```

Pistemääriä -4 ja 100 ei siis huomioida tuloksissa.

3. Metodeja (6 p)

   (a) (3p) Tee metodi `public static void tulostaVali(int reuna1, int reuna2)` joka tulostaa kasvavassa suuruusjärjestyksessä jokaisen parametriensa määrittelemällä välillä olevan kokonaisluvun.

   Esim. jos kutsutaan `tulostaVali(3, 7)` tulostuu

   ```
   3 4 5 6 7
   ```

   Metodi toimii myös jos ensimmäisenä oleva parametri on toista suurempi, eli jos kutsutaan `tulostaVali(10, 8)` tulostuu

   ```
   8 9 10
   ```

   Luvut siis tulostetaan *aina kasvavassa suuruusjärjestyksessä* riippumatta siitä kumpi metodin parametreistä on suurempi, ensimmäinen vaiko toinen.

   (b) (3p) Tee metodi `public static boolean kummatkinLoytyy(int[] luvut, int luku1, luku2)` joka saa parametrikseen kokonaislukutaulukon ja kaksi kokonaislukua. Metodi palauttaa `true` jos *molemmat* sen parametreinä saamista luvuista (`luku1` ja `luku2`) löytyvät metodin parametrina olevasta taulukosta. Muissa tilanteissa metodi palauttaa `false`.

   Esim. saadessaan parametriksi taulukon `[1,5,3,7,5,4]` ja luvut `5` ja `7`, metodi palauttaa `true`. Jos metodi saisi parametriksi taulukon `[1,5,3,2]` ja luvut `7` ja `3` palauttaisi metodi `false`.

   Tee myös pääohjelma, joka demonstroi metodin käyttöä.

   Huom: jos et osaa käyttää taulukkoja, voit tehdä metodista version `public static boolean kummatkinLoytyy(ArrayList<Integer> luvut, int luku1, int luku2)` jossa se siis saa parametrikseen luvut sisältävän ArrayList:in sekä etsittävät luvut.

4. Olio-ohjelmointi, osa 1 (6p)

   Tee luokka `Varasto`. Varastolla on kapasiteetti, joka on kokonaisluku, myös varastoon talletettuna oleva tavaroiden määrä ilmaistaan kokonaislukuna. Varaston kapasitetti määritellään konstruktorin parametrin avulla (voit olettaa että parametrin arvo on positiivinen). Luokalla on seuraavat metodit:

   - `void lisaa(int maara)`, joka lisää varastoon parametrin ilmoittaman määrän tavaroita. Jos määrä on negatiivinen, varaston tila ei muutu. Varaston tavaramäärä ei saa lisäyksen yhteydessä kasvaa kapasitettia suuremmaksi. Jos lisättävä määrä ei mahdu kokonaan varastoon, täyttyy varasto ja loput lisättävät 'häviävät'.
   - `int tilaa()`, joka palauttaa varastossa olevan tyhjän tilan määrän.
   - `void tyhjennä()`, joka tyhjentää varaston.
   - `toString()`, joka palauttaa varaston tilanteen tekstiesityksen allaolevan esimerkin tapaan muotoiltuna, huomaa tilanne jossa varasto on tyhjä!

   Seuraavassa esimerkki, joka demonstroi oikein toteutetun varaston toimintaa:

```
public static void main(String[] args) {
    Varasto varasto = new Varasto(24);
    varasto.lisaa(10);
    System.out.println(varasto);
    System.out.println("varastossa tilaa "+varasto.tilaa());
    varasto.lisaa(-2);
    System.out.println(varasto);
    varasto.lisaa(50);
    System.out.println(varasto);
    varasto.tyhjenna();
    System.out.println(varasto);
}
```

   jos luokka on toteutettu oikein, tulostuu

```
kapasitetti: 24 tavaraa 10
varastossa tilaa 14
kapasitetti: 24 tavaraa 10
kapasitetti: 24 tavaraa 24
kapasitetti: 24 tyhjä
```

5. Olio-ohjelmointi, osa 2 (6p)

   Tehtävässä tehdään ohjelma kirjahyllyn sisällön hallintaan. Käytössä on valmis luokka `Kirja`:

```
public class Kirja {
    private String kirjailija;
    private String nimi;

    public Kirja(String kirjailija, String nimi) {
        this.kirjailija = kirjailija;
        this.nimi = nimi;
    }

    public String getKirjailija() {
        return this.kirjailija;
    }
```

```
    @Override
    public String toString() {
        return this.kirjailija + ": " + this.nimi;
    }
}
```

Tehtävänä on ohjelmoida luokka `Kirjahylly`, joka toimii seuraavan esimerkin kuvaavalla tavalla:

```
public static void main(String[] args) {
    Kirjahylly hylly = new Kirjahylly();

    hylly.lisaaKirja("Kent Beck", "Test Driven Development");
    hylly.lisaaKirja("Kent Beck", "Extreme Programming Embraced");
    hylly.lisaaKirja("Martin Fowler", "UML Distilled");
    hylly.lisaaKirja("Fedor Dostoyevski", "Crime and Punishment");

    hylly.tulosta();
    System.out.println("---");

    hylly.hae("Kent Beck");
}
```

kun luokka on toteutettu oikein, tulostuu

```
kirjoja yhteensä 4
kirjat:
 Kent Beck: Test Driven Development
 Kent Beck: Extreme Programming Embraced
 Martin Fowler: UML Distilled
 Fedor Dostoyevski: Crime and Punishment
---
löytyi:
 Kent Beck: Test Driven Development
 Kent Beck: Extreme Programming Embraced
```

Luokan tulee tallettaa hyllyyn lisätyt kirjat `Kirja`-olioita sisältävään ArrayListiin!

Kuten esimerkistä näemme, luokalle `Kirjahylly` tulee toteuttaa seuraavat metodit:

- `void lisaaKirja(String kirjailija, String nimi)`, joka lisää hyllyyn kirjan, jolla on parametrin kuvaama kirjailija ja nimi.
- `void tulosta()`, joka tulostaa kirjahyllyn tiedot yo. esimerkin tapaan muotoiltuna
- `hae(String kirjailija)`, joka tulostaa hyllyssä olevista kirjoista ne, joiden kirjailija on metodin parametrina määritelty, tulostusasun tulee olla samanlainen kuten yo. esimerkissä.

ArrayListin käyttöohje

- Konstruktori `public ArrayList<T>()` luo uuden `ArrayList`-olion, jossa listan alkiot ovat tyyppiä `T`.
- `public boolean add(T x)` lisää listan loppuun olion `x`.
- `public int get(int i)` palauttaa listan alkion kohdasta `i`.