Välikoe / 26.3

Vastaa neljään (4) tehtävään ja halutessa bonustehtäviin B1 ja/tai B2, (tuovat lisäpisteitä). Bonustehtävät saa tehdä vaikkei olisi tehnyt siihen tehtävään liittyvää tehtävää. Aikaa 4 tuntia. **Jokainen vastaus omalle paperilleen**, bonustehtävät voi kirjoittaa samalle. Tehtävistä 1-6 SAA vastata vain 4:ään. Jos vastaa useampaan, niin 4 **HUONOINTA** arvostellaan.

1. Keskimaassa kuhisee. Selvitä pöytätestillä kuka on Keski-Maan ykkösvelho. Täytä liitteenä annettava pohja, jota on täytetty hieman alkuun malliksi. (6p)

```
- Merkitse harmaaksi ne alueet, jolloin muuttujaa ei ole olemassa.
- Ruutuun merkintä vain jos muuttujan arvo muuttuu tai olio muuttuu
  roskaksi.
- Merkitse iso R-kirjain kun olio muuttuu roskaksi.
- Merkitse * jokaisen muuttujan päälle joka on viitemuuttuja.
- N1 tarkoittaa ensimmäistä kekoon luotua oliota. N2 toista jne.
- Käytä &-merkkiä olioviitteisiin. (esim. &N1 viittaa N1:een)
- Aikasi säästämiseksi sinun ei tarvitse kirjoittaa ohjelman koko
  riviä, vaan pelkkä rivinumero riittää
- Ole tarkkana sillä koodissa on samanimisiä muuttujia eri rooleissa
- Huomaa merkitä myös aliohjelmien/metodien lokaalien
  muuttujien muutokset.

/*01*/public class KeskiMaa2 {
/*02*/
/*03*/      public static class Velho {
/*04*/
/*05*/            private int mana;
/*06*/            private int kunto;
/*07*/
/*08*/            public Velho(int mana, int kunto) {
/*09*/                  this.mana = mana;
/*10*/                  this.kunto = kunto;
/*11*/            }
/*12*/
/*13*/            public void teeLoitsu(Velho velho, int voimakkuus) {
/*14*/                  if(mana >= voimakkuus) {
/*15*/                        this.mana -= voimakkuus;
/*16*/                        velho.osuma(voimakkuus);
/*17*/                  }
/*18*/            }
/*19*/
/*20*/            public void osuma(int voimakkuus) {
/*21*/                  this.kunto -= voimakkuus;
/*22*/                  System.out.println("Osuma voimalla " +
                                          voimakkuus);
/*23*/            }
/*24*/
/*25*/            public boolean onKuollut() {
/*26*/                  return (kunto <= 0);
/*27*/            }
/*28*/
/*29*/      }
/*30*/
/*31*/      public static void main(String[] args) {
```

```
/*32*/            Velho gandalf = new Velho(5,10);
/*33*/            Velho saruman = new Velho(5,5);
/*34*/
/*35*/            saruman.teeLoitsu(gandalf, 4);
/*36*/
/*37*/            Velho merlin = saruman;
/*38*/
/*39*/            merlin.teeLoitsu(gandalf, 5);
/*40*/
/*41*/            if(gandalf.onKuollut()) {
/*42*/                  System.out.println("Gandalf kuoli");
/*43*/                  gandalf = null;
/*44*/            } else {
/*45*/                  gandalf.teeLoitsu(saruman,5);
/*46*/                  gandalf = merlin;
/*47*/            }
/*48*/
/*49*/            if(saruman.onKuollut()) {
/*50*/                  System.out.println("Saruman kuoli");
/*51*/                  saruman = null;
/*52*/            }
/*53*/      }
/*54*/}
```

2. a) On annettu kaksi kokonaislukujonoa. Kirjoita sanallisesti tarkka algoritmi, joka laskee ensimmäisestä jonosta kaikkien jälkimmäisen jonon esiintymien lukumäärän. Alla `ComTest` -esimerkki "käytöksestä". Jos jotakin esimerkkitapauksia puuttuu, "määrittele" niiden toiminta vastaavasti. 3p.

```
* @example
* <pre name="test">
*   int [] t1 = {1, 2, 3, 1, 2, 3};
*   int [] t2 = {1, 2};
*   int [] t3 = {3, 1};
*   laskeEsiintymat(t1,t2) === 2;
*   laskeEsiintymat(t1,t3) === 1;
* </pre>
```

b) Toteuta a)-kohdan algoritmi Javalla. (3p)

3. a) Muodosta luokka `Opiskelija` ja täydennä luokkaa `Ainejarjesto` siten, että alla oleva pääohjelma toimii. (4p)

b) Piirrä kuva ohjelman tietorakenteesta pyydetyissä kohdissa (2p)

```
public interface Jasen {
        public String annaNimesi();
}

public class Ainejarjesto implements Jasen {

        private Jasen[] jasenet;

        /**
         *Testipääohjelma
         *@param args ei käytössä
         */
```

```
        public static void main(String[] args) {
                Ainejarjesto Linksutin = new Ainejarjesto("Linksutin");
                Ainejarjesto miinus = new Ainejarjesto("Miinus");

                //Piirrä kuva tietorakenteesta
                System.out.println(Linksutin);

                Linksutin.lisaaJasen(new Opiskelija("Mikko"));
                Linksutin.lisaaJasen(new Opiskelija("Ansku"));
                Linksutin.lisaaJasen(new Opiskelija("Sauli"));
                Linksutin.lisaaJasen(new Opiskelija("Eetu"));
                Linksutin.lisaaJasen(miinus);

                //Piirrä kuva tietorakenteesta
                System.out.println(Linksutin);
        }
}
```

yllä oleva pääohjelma tulostaa:

```
Linksutin
jäseniä 0

Linksutin
jäseniä 5
Mikko
Ansku
Sauli
Eetu
Miinus
```

4. a) Mitä seuraava aliohjelma yrittää tehdä? (2p)

```
public static int[ ][ ] teeJuttu(int[ ][ ] a, int b) {
    int[ ][ ] c;
    for (int i; i<a.length; i++) for (int j; j<a.length; j++) c[j][i]
    = a[j][i] * b; return c;
}
```

b) Mitä siinä on mielestäsi vialla, jos mitään? (2p)

c) Toteuta itse vastaava metodi, mutta paremmin. Perustele (lyhyesti) miksi oma tapasi on parempi. (2p)

5. Toteuta metodit, joiden avulla voidaan lukea tekstitiedostosta muotoa

```
1|1|1
2|1|2
6|6|6
```

oleva kokonaislukumatriisi, kertoa se kokonaisluvulla ja kirjoittaa uusi matriisi tiedostoon. Testiohjelman

```
        int[ ][ ] matriisi = lueMatriisi("matriisi.txt");
```

```
matriisi = kerroMatriisiKokonaisluvulla(matriisi, 2);
kirjoitaMatriisi(matriisi, "matriisi2.txt");
```

tulisi tuottaa `matriisi2.txt` -niminen tiedosto, jonka sisältö olisi seuraavanlainen:

```
2|2|2
4|2|4
12|12|12
```

Määrittele itse mahdolliset poikkeustapaukset ja niiden käytös. (6p)

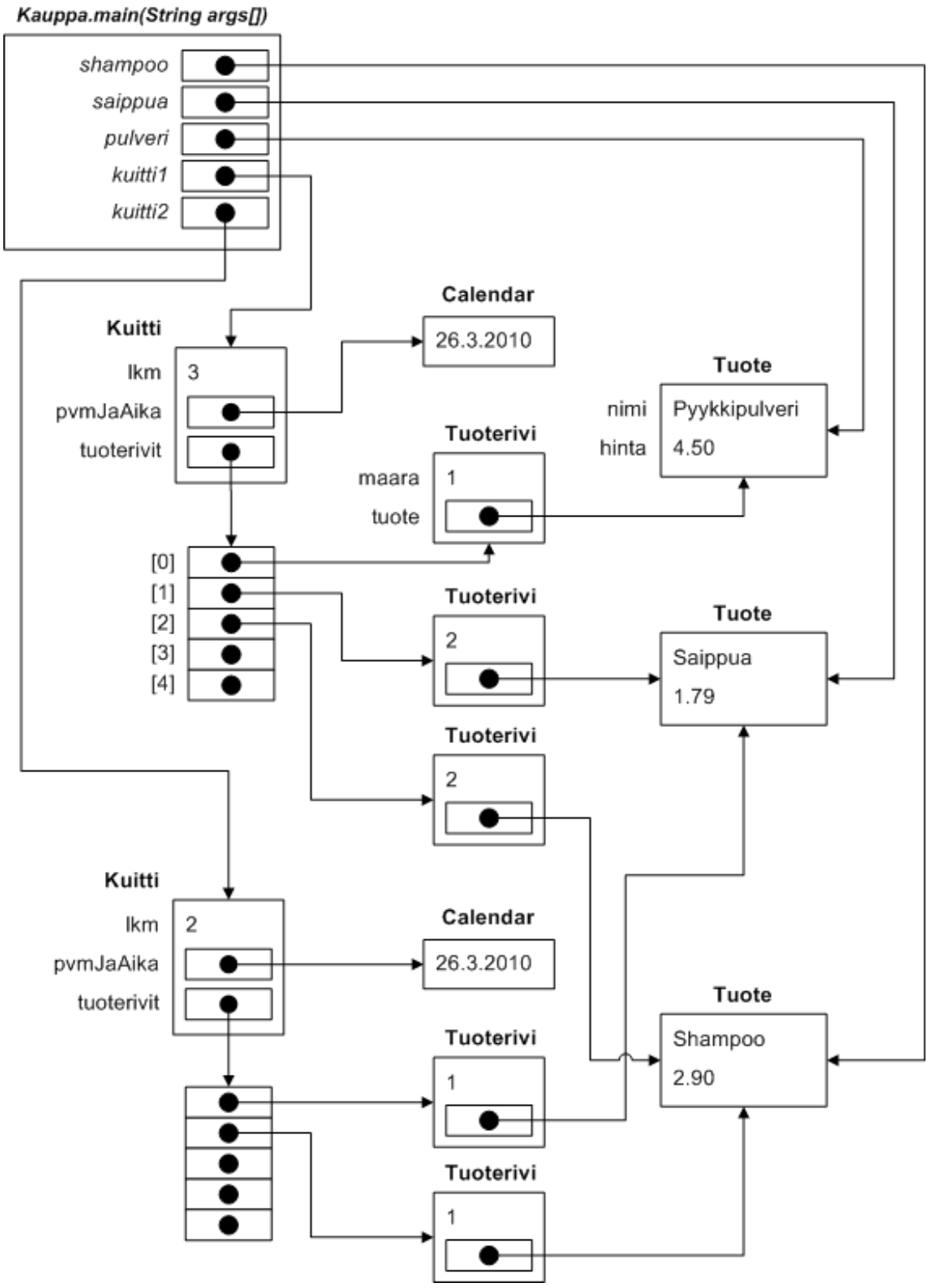
6. Liitteessä on kuva ja neljän eri luokan osittainen Java-koodi. Täydennä kommentoidut kohdat (6 kpl) siten, että saat aikaan kuvassa esitetyt oliot ja viitteet olioiden välille. "Merkitse vastaukseesi selvästi täydentämiesi kohtien numerot. (6p)

B1. Kerro onko alla olevista lauseista joku oikein/väärin ja miksi (`Jasen`-luokan sisältö ei juurikaan vaikuta kysymyksen asetteluun, oletetaan sen kuitenkin olevan luokka, ei rajapinta ja sisällöltään vähän kuten kurssin malliohjelmassa) (2p):

```
Jasen jasenet[] = (Jasen)(new Object[10]);
Jasen jasenet[] = (Jasen [])(new Object[10]);
Jasen jasen = (Jasen)(new Object());
```

B2. Mitä tarkoittaa Javassa `extends` ja mitä `implements`. Molemmilla voidaan toteuttaa mm. polymorfismi, mutta mikä niillä on erona koodin kirjoittamisen kannalta? (1p)

Liite 1


Kauppa.main(String args[])
shampoo
saippua
pulveri
kuitti1
kuitti2
Kuitti
lkm
3
pvmJaAika
tuoterivit
Calendar
26.3.2010
Tuote
nimi
Pyykkipulveri
hinta
4.50
Tuoterivi
maara
1
tuote
[0]
[1]
[2]
[3]
[4]
Tuoterivi
2
Tuote
Saippua
1.79
Tuoterivi
2
Kuitti
lkm
2
pvmJaAika
tuoterivit
Calendar
26.3.2010
Tuote
Shampoo
2.90
Tuoterivi
1
Tuoterivi
1

Kauppa.java

```
/**
 * Testaa Kuitti- ja Tuote-luokkia.
 * @author Ville Lahtinen
 */
public class Kauppa {

	/**
	 * Testaa Kuitti- ja Tuote-luokkia.
	 * @param args ei käytössä
	 */
	public static void main(String[] args) {
		Tuote shampoo;
		Tuote saippua;
		Tuote pulveri;
		Kuitti kuitti1;
		Kuitti kuitti2;

// KOHTA 1: Täydennä tarvittava ohjelmakoodi.

		kuitti1.lisaa(shampoo, 1);
		kuitti1.lisaa(shampoo, 1);
		kuitti1.lisaa(saippua, 2);
		kuitti1.lisaa(pulveri, 1);
	}

}
```

Kuitti.java

```
import java.util.Calendar;

/**
 * Kuitti, joka säilyttää tuoterivejä
 * @author Ville Lahtinen
 */
public class Kuitti {
	private static int KASVATUS = 5;
	private Tuoterivi[] tuoterivit = new Tuoterivi[KASVATUS];
	private int lkm = 0;
	private Calendar pvmJaAika = Calendar.getInstance();

	/**
	 * Etsii tietorakenteesta tuoterivin, jolla on tuote.
	 * @param tuote tuote, jolla etsitään
	 * @return tuoterivi, jolla on tuote. Jos riviä ei löydy,
	 * palautetaan null.
	 */
	private Tuoterivi etsiTuoterivi(Tuote tuote) {
// KOHTA 2: Täydennä tarvittava ohjelmakoodi.
	}

	/**
	 * Kasvattaa tietorakenteen kokoa.
	 */
	private void kasvata() {
		Tuoterivi[] uusi = new Tuoterivi[tuoterivit.length +
```

```
KASVATUS];
            for (int i = 0; i < lkm; i++) {
                  uusi[i] = tuoterivit[i];
            }
            tuoterivit = uusi;
      }

// KOHTA 3: Kirjoita JavaDoc-kommentti.
      public void lisaa(Tuote tuote, int maara) {
// KOHTA 4: Täydennä tarvittava ohjelmakoodi.
      }

      /**
       * Lisää tuoterivin tietorakenteeseen.
       * @param tuoterivi lisättävä tuoterivi
       */
      private void lisaa(Tuoterivi tuoterivi) {
            if (lkm >= tuoterivit.length) {
                  kasvata();
            }
            tuoterivit[lkm] = tuoterivi;
            lkm++;
      }
}
```

Tuoterivi.java

```
/**
 * Säilyttää tuotetietoa ja kappalemäärää.
 * @author Ville Lahtinen
 */
public class Tuoterivi {

      private Tuote tuote;
      private int maara;

      /**
       * Alustaa uuden tuoterivin.
       * @param tuote tuote, jonka tiedot rivi säilyttää
       * @param maara tuotteen määrä
       */
      public Tuoterivi(Tuote tuote, int maara) {
            this.tuote = tuote;
            this.maara = maara;
      }

      /**
       * Kertoo, liittyykö tuoterivi tuotteeseen.
       * @param tuote tuote, johon verrataan
       * @return true, jos tuoterivi liittyy tuotteeseen, muuten false
       */
      public boolean omistatko(Tuote tuote) {
// KOHTA 5: Täydennä tarvittava ohjelmakoodi.
      }

      /**
       * Lisää tuotteen kappalemäärää.
       * @param maara lisättävä määrä
       */
```

```
	public void lisaaMaaraa(int maara) {
		this.maara += maara;
	}
}
```

Tuote.java

```
/**
 * Säilyttää tuotteen nimeä ja hintaa.
 * @author Ville Lahtinen
 */
public class Tuote {
	private String nimi;
	private double hinta;

	/**
	 * Alustaa uuden tuotteen.
	 * @param nimi tuotteen nimi
	 * @param hinta tuotteen kappalehinta
	 */
	public Tuote(String nimi, double hinta) {
		this.nimi = nimi;
		this.hinta = hinta;
	}

// KOHTA 6: Lisää tarvittava metodi (1 kpl).
// Muista myös JavaDoc-kommentti!
}
```