DAT100

Øving 6: Obligatorisk øving 3
9. oktober 2014
Gruppe:
145157 Pål V. Gjerde
145161 Henriette Lie

```java
/**
 * Besvarelse på oppgave 1
 * @author Pål V. Gjerde
 * @author Henriette Lie
 *
 */
public class Oppgave1 {

    /**
     * Tar en streng og returnerer strengen baklengs
     * @param inn Strengen som skal vendes om
     * @return Den oppgitte strengen baklengs
     */
    public static String baklengs(String inn) {
        // Opprett en tom StringBuilder
        StringBuilder ut = new StringBuilder();

        // Les gjennom strengen i omvendt rekkefølge (fra slutt til begynnelse)
        for (int i = inn.length()-1; i >= 0; --i) {
            // Legg til karakteren til StringBuilderen
            ut.append(inn.charAt(i));
        }

        // Returnerer strengen som ble bygget
        return ut.toString();
    }

    /**
     * Sjekker om en integerarray er sortert fra lavest til høyest
     * @param inn En integerarray å sjekke
     * @return true hvis arrayen er sortert, false hvis ikke
     */
    public static boolean erSortert(int[] inn) {
        /*
         * Kontroller at alle tallene i arrayen er større eller likt tallet før.
         * Det første tallet kontrolleres ikke siden det ikke har et tall før.
         * Hvis det finnes et tall som er mindre enn tallet før, returner false.
         */
        for (int i = 1; i < inn.length; ++i) {
            if (inn[i] < inn[i-1]) return false;
        }

        // Arrayen er sortert - returner true
        return true;
    }

    /**
     * Returnerer den strengen i en array som er først i Unicode-rekkefølge
     * @param inn En array av strenger
     * @return Strengen som er først i Unicode-rekkefølge
     */
    public static String forstIUnicode(String[] inn) {
        // Start med den første strengen
        String ut = inn[0];

        // Sjekk alle strengene
        for (String enStreng : inn) {
            // Hvis en streng er før den vi har nå, bytt den vi har med den nye.
            if (enStreng.compareTo(ut) < 0) {
                ut = enStreng;
            }
        }

        // Returner den strengen vi fant
```

```
        return ut;
    }

    /**
     * Sammenligner to strenger for å se hvilken som kommer først i
     * Unicode-rekkefølge.
     * @param en Den første strengen
     * @param to Den andre strengen
     * @return Et negativt tall hvis den første strengen kommer før den andre;
     *      et positivt tall hvis den andre kommer før den første; 0 hvis
     *      strengene er like
     */
    public static int sammenlignStrenger(String en, String to) {
        /*
         * Sammenlign hvert tegn i strengene (stopp hvis en streng går tom for
         * tegn).
         */
        for (int i = 0; i < Math.min(en.length(), to.length()); ++i) {
            /*
             * Hvis strengene har forskjellige tegn på denne plassen, er de
             * forskjellige, og forskjellen mellom tegnene kan returneres som
             * forskjellen mellom strengene (siden det nøyaktige tallet ikke
             * har noe å si, kun positivt/negativt/0).
             */
            if (en.charAt(i) != to.charAt(i)) {
                return en.charAt(i) - to.charAt(i);
            }
        }

        /*
         *  Hvis ingen ulike tegn fantes, returner forskjellen mellom lengdene
         *  på strengene. Hvis strengene er like vil dette være 0.
         */
        return en.length() - to.length();
    }

    public static void main(String[] args) {
        System.out.println("Teststreng baklengs: " + baklengs("Teststreng"));
        int[] t1 = {1, 2, 4, 6, 9};
        System.out.println("1, 2, 4, 6, 9 er sortert: " + erSortert(t1));
        int[] t2 = {1, 3, 2, 4, 7};
        System.out.println("1, 3, 2, 4, 7 er sortert: " + erSortert(t2));
        String[] t3 = {"En", "To", "Fire", "Seks", "Ni"};
        System.out.print("Først av En, To, Fire, Seks, Ni: ");
        System.out.println(forstIUnicode(t3));
        String[] t4 = {"Erlend", "Anne", "Nora", "Aila"};
        System.out.print("Først av Erlend, Anne, Nora, Aila: ");
        System.out.println(forstIUnicode(t4));
        System.out.print("Sammenlign 'Streng' og 'Streng2': ");
        System.out.println(sammenlignStrenger("Streng", "Streng2"));
        System.out.print("Sammenlign 'Streng2' og 'Streng': ");
        System.out.println(sammenlignStrenger("Streng2", "Streng"));
        System.out.print("Sammenlign 'Streng' og 'Streng': ");
        System.out.println(sammenlignStrenger("Streng", "Streng"));
    }
}
```

Denne oppgaven består av fire uavhengige metoder.
Metoden som returnerer en string baklengs gjør det ved å opprette en StringBuilder, og lese gjennom stringen som skal snus baklengs, og lagre ett og ett tegn. Til slutt returneres resultatet.
Metoden som undersøker om tallene i et array er sortert stigende. Det gjøres ved å

undersøke at hvert tall etter det første er høyere eller likt tallet før. Hvis et tall finnes som er mindre enn tallet før, er tabellen ikke sortert.
Metoden som finner den strengen som kommer alfabetisk først etter Unicode-rekkefølge starter med den første strengen, og sammenligner den med hver av de andre strengene. Hvis en tidligere streng finnes, byttes den første strengen ut.
Bonusmetoden som sammenligner to strenger sammenligner hvert tegn i den korteste strengen med tegnet i samme posisjon i den andre. Hvis de er ulike, returneres forskjellen mellom de tegnene. Hvis ingen slike tegn finnes, returneres forskjellen mellom lengene på strengen – som er 0 hvis strengene er like.

Utskrift:

```
Teststreng baklengs: gnertstseT
1, 2, 4, 6, 9 er sortert: true
1, 3, 2, 4, 7 er sortert: false
Først av En, To, Fire, Seks, Ni: En
Først av Erlend, Anne, Nora, Aila: Aila
Sammenlign 'Streng' og 'Streng2': -1
Sammenlign 'Streng2' og 'Streng': 1
Sammenlign 'Streng' og 'Streng': 0
```

```java
import java.util.Arrays;
import java.util.Random;
import easyIO.Out;

/**
 * Besvarelse på oppgave 2
 * @author Pål V. Gjerde
 * @author Henriette Lie
 *
 */
public class Oppgave2 {
    private static Out out = new Out();
    private static Random rnd = new Random();

    /**
     * Måler tiden det tar å fylle en integerarray med tilfeldige tall (valgt
     * fra alle mulige integerverdier), og sortere den med Arrays.sort().
     * @param antallElementer Antall elementer arrayen skal inneholde
     * @return Tiden sorteringen tok, målt i millisekunder.
     */
    public static int malTid(int antallElementer) {
        // Opprett en array og fyll den med tilfeldige verdier
        int[] tilfeldig = new int[antallElementer];
        for (int i=0; i<tilfeldig.length; ++i) {
            tilfeldig[i] = rnd.nextInt();
        }

        // Les tid før sortering
        long tid1 = System.nanoTime();

        // Sorter array
        Arrays.sort(tilfeldig);

        // Les tid etter sortering, regn ut forskjell
        long tid2 = System.nanoTime();
        int forskjell = (int)((tid2 - tid1) / 1000000);

        // Returner forskjellen
        return forskjell;
    }

    /**
     * Måler hvor lang tid det tar å sortere integerarray med forskjellige
     * størrelser som er fylt med tilfeldige tall. Måler 20 ganger for hver
     * størrelse, skriver ut resultatet for hver måling samt gjennomsnittet
     * for alle 20 forsøkene med den størrelsen.
     * @param args Kreves for main-metoden, men brukes ikke.
     */
    public static void main(String[] args) {
        int[] antallElementer = {
            1000000,
            5000000,
            10000000,
            100000000
        };

        int antallForsok = 20;

        // For hvert antall som skal prøves:
        for (int antall : antallElementer) {
            int sum = 0;

            // Skriv ut overskriften
            out.outln(antall + " elementer:");
```

```
            out.out("Forsøk", 8, Out.LEFT);
            out.outln("Tid", 8, Out.LEFT);
            out.outln("----------------");

            // Prøv 20 ganger
            for (int i=0; i < antallForsok; ++i) {
                // Mål tid, og skriv ut
                int tid = malTid(antall);
                sum += tid;
                out.out(i+1, 8);
                out.outln(tid + "ms", 8, Out.RIGHT);
            }

            // Regn ut og skriv ut gjennomsnittet av forsøkene
            out.outln("----------------");
            out.outln("Gjennomsnittlig tid: " + (sum/antallForsok) + "ms");
            out.outln();
        }
    }
}
```

Denne klassen måler tiden det tar å sortere et array fylt med tilfeldige integre. Fire forskjellige størrelser brukes for å måle, og hver størrelse måles 20 ganger. I tillegg til resultatet av hver måling skrives gjennomsnittet av alle 20 ut.
En egen metode brukes til å måle tiden gitt en arraystørrelse. Tiden som blir brukt måles ved hjelp av System.nanoTime(), som returnerer tiden i nanosekunder (dette blir omgjort til millisekunder før det returneres).
Tallene som brukes velges av metoden nextInt() i Random-klassen. Denne velger tallene fra alle integerverdier Java kan representere.
Målingene vil være avhengig av maskinen koden kjøres på. Dette kan anses som en svakhet, men vil i de fleste tilfeller være en fordel, da denne koden kan brukes til å sammenligne ulike maskiners evne til å kjøre Java.

Utskrift:

```
1000000 elementer:
Forsøk  Tid
----------------
       1   244ms
       2   247ms
       3   175ms
       4   177ms
       5   173ms
       6   173ms
       7   190ms
       8   170ms
       9   178ms
      10   174ms
      11   172ms
      12   190ms
      13   173ms
      14   173ms
      15   173ms
      16   172ms
      17   174ms
      18   174ms
      19   172ms
      20   173ms
----------------
```

Gjennomsnittlig tid: 182ms

5000000 elementer:

Forsøk	Tid
1	1081ms
2	958ms
3	977ms
4	968ms
5	983ms
6	1018ms
7	977ms
8	960ms
9	979ms
10	990ms
11	993ms
12	1021ms
13	978ms
14	994ms
15	1049ms
16	999ms
17	982ms
18	968ms
19	998ms
20	1032ms

Gjennomsnittlig tid: 995ms

10000000 elementer:

Forsøk	Tid
1	2054ms
2	2087ms
3	2086ms
4	2134ms
5	2326ms
6	2146ms
7	2460ms
8	2112ms
9	2138ms
10	2016ms
11	2084ms
12	2109ms
13	2122ms
14	2033ms
15	2082ms
16	2196ms
17	2074ms
18	2082ms
19	2056ms
20	2085ms

Gjennomsnittlig tid: 2124ms

100000000 elementer:

Forsøk	Tid
1	24656ms
2	24269ms
3	24260ms
4	24040ms
5	24148ms
6	24399ms
7	24616ms

```
       8 28358ms
       9 23500ms
      10 23604ms
      11 23248ms
      12 23629ms
      13 26293ms
      14 27605ms
      15 33568ms
      16 26444ms
      17 25841ms
      18 25106ms
      19 23776ms
      20 23699ms
----------------
Gjennomsnittlig tid: 25252ms
```

```java
import java.util.Arrays;
import java.util.Random;
import easyIO.Out;

/**
 * Besvarelse på oppgave 3
 * Programmet kaster først 100 terninger. Deretter teller det forekomsten av
 * hvert tall, og gjennomsnittet av alle tallene. Så skriver det ut antall kast
 * det tok for å få en sekser, og hvilket tall som forekom oftest.
 * @author Pål V. Gjerde
 * @author Henriette Lie
 *
 */
public class Oppgave3 {
    private static Out out = new Out();
    private static Random rnd = new Random();

    // Antall kast
    public static final int ANTALL_KAST = 100;

    public static void main(String[] args) {
        // Arrays for alle kast og antall forekomster av hvert tall
        int[] kast = new int[ANTALL_KAST];
        int[] antall = new int[6];
        Arrays.fill(antall, 0);

        // Tellere for antall kast før en sekser kom, og summen av tallene
        int kastForSekser = 0;
        int sum = 0;

        // Overskrift for kastene
        out.outln("Kast:");

        for (int i=0; i < ANTALL_KAST; ++i) {
            // Kast en terning, lagre kastet i arrayet for kast, og legg til en
            // til antallet av det resultatet
            int detteKastet = rnd.nextInt(6)+1;
            kast[i] = detteKastet;
            ++antall[detteKastet-1];

            // Hvis dette er den første sekseren, lagre antall kast det tok
            // å få den
            if (kastForSekser == 0 && detteKastet == 6) {
                kastForSekser = i+1;
            }

            // Øk summen og skriv ut kastet. Skriv ut ny linje hvis 25 kast
            // har blitt skrevet ut
            sum += detteKastet;
            out.out(detteKastet, 2, Out.LEFT);
            if ((i+1) % 25 == 0) out.outln();
        }

        // Skriv ut antall kast for hvert tall
        out.outln();
        out.outln("Antall kast per tall:");
        out.outln("---------------------");
        for (int i = 0; i < 6; ++i) {
            out.out(i+1, 5);
            out.outln(antall[i], 8);
        }
        out.outln();

        // Skriv ut gjennomsnittet (som et desimaltall med 3 desimaler)
```

```
        out.out("Gjennomsnitt av kastet: ");
        out.outln(sum / (float)ANTALL_KAST, 3);

        // Skriv ut antall kast før en sekser ble funnet
        out.out("Antall kast før første sekser: ");
        out.outln(kastForSekser);

        // Finn ut hvilket tall forekom oftest
        int hoyesteAntall = 0;
        int oftesteKast = 0;
        for (int i = 0; i < 6; ++i) {
            if (antall[i] > hoyesteAntall) {
                hoyesteAntall = antall[i];
                oftesteKast = i+1;
            }
        }

        // Skriv ut tallet som forekom oftest
        out.out("Tallet som forekom oftest: ");
        out.outln(oftesteKast);
    }
}
```

Denne koden kaster 100 terninger, skriver ut alle kastene, teller opp forekomstene av hvert tall, regner ut gjennomsnittet, teller hvor mange kast det tok å få den første sekseren, og skriver ut hvilket tall som forekom oftest.
Det meste av dette foregår i en enkel for-løkke. Først gjøres et kast som lagres i et array. Deretter inkrementerer koden antall forekomster av tallet som kom opp. Hvis tallet er den første sekseren (antall kast før 6 er 0 hvis ingen har forekommet) lagres antallet kast som har blitt gjort. Så legges resultatet til summen, og skrives ut.
Etter dette kjører en kort loop over antallet forekomster av hvert tall og skriver dem ut, så regnes gjennomsnittet ut fra summen, og antall kast før første sekser skrives ut. Så finnes det tallet med den høyeste forekomsten, og det skrives ut.
En svakhet er kjent: koden tar ikke høyde for muligheten for 100 kast uten seksere. Dette krasjer ikke programmet, men vil skrive ut 0 kast før første sekser.

Utskrift:

```
Kast:
2 4 6 2 3 6 1 6 2 6 2 2 5 5 5 3 3 3 3 2 1 6 3 3 3
1 2 1 4 6 6 1 4 4 6 1 4 6 1 2 3 2 5 2 5 2 2 6 6 1
4 4 1 1 5 3 2 4 3 2 3 6 6 2 3 4 2 2 5 2 2 5 1 2 4
2 1 6 3 5 2 1 1 3 6 3 3 2 6 1 6 2 6 4 4 2 5 6 1 4

Antall kast per tall:
---------------------
    1      16
    2      25
    3      17
    4      13
    5      10
    6      19

Gjennomsnitt av kastet: 3.330
Antall kast før første sekser: 3
Tallet som forekom oftest: 2
```

```java
/**
 * Besvarelse på oppgave 4
 * @author Pål V. Gjerde
 * @author Henriette Lie
 *
 */
public class Vare {
    // Objektvariabler
    private int varenr;
    private String navn;
    private double pris;

    // Konstanter for moms og utskrift av vare
    private static final double MOMS = 0.2;
    private static final String UTFORMAT = "#%d %s (kr %.2f)";

    /**
     * Konstruktør uten parametre
     * Oppretter en "navnløs vare" med varenr og pris 0
     */
    public Vare() {
        this(0, "Navnløs vare", 0.0);
    }

    /**
     * Konstruktør med parametre
     * @param varenr Varenummeret til varen
     * @param navn Navnet til varen
     * @param pris Hva varen koster
     */
    public Vare (int varenr, String navn, double pris) {
        this.varenr = varenr;
        this.navn = navn;
        this.pris = pris;
    }

    /*
     * Get/set-metoder
     */
    public int getVarenr() { return varenr; }
    public void setVarenr(int varenr) { this.varenr = varenr; }
    public String getNavn() { return navn; }
    public void setNavn(String navn) { this.navn = navn; }
    public double getPris() { return pris; }
    public void setPris(double pris) { this.pris = pris; }

    /**
     * Regner ut momsen (merverdiavgift) på varen, rundet ned til to desimaler
     * @return Momsen på varen
     */
    public double moms() {
        return (int)(pris*MOMS*100) / 100.0;
    }

    /**
     * Sjekker om denne varen er billigere enn en annen vare
     * @param v Den andre varen
     * @return true hvis denne varen er billigere, false hvis ikke
     */
    public boolean billigereEnn(Vare v) {
        return this.pris < v.pris;
    }

    /**
```

```
     * Skriver ut varens informasjon til skjermen
     */
    public void skriv() {
        System.out.println(toString());
    }

    /**
     * Returnerer varens informasjon i stringform
     */
    public String toString() {
        return String.format(UTFORMAT, varenr, navn, pris);
    }

    public static void main(String[] args) {
        Vare v1 = new Vare(1, "Testvare 1", 150);
        Vare v2 = new Vare(2, "Testvare 2", 220);
        System.out.print("Vare 1: ");
        v1.skriv();
        System.out.print("Vare 2: ");
        v2.skriv();
        System.out.println("Moms på vare 1: "+v1.moms());
        System.out.println("Moms på vare 2: "+v2.moms());
        System.out.println("Vare 1 billigere: "+v1.billigereEnn(v2));
    }
}
```

Dette er en generisk vare-klasse. Den lagrer varenummer, navn og pris for en vare. I tillegg til disse objektvariablene, har vi brukt to statiske konstanter – en for moms og en for utskriftsformatet til varen.
Klassen har get- og set-metoder for alle objektvariablene.
Moms-metoden regner ut momsen av en vare, som er definert som 20% av prisen. Den åpenbare måten å gjøre dette er å gange prisen med momsen, men vi har gått for en litt mer komplisert måte – prisen ganges med 100 og castes til en integer, og deles deretter på 100,0. Dette er fordi den åpenbare måten kan føre til flyttallspresisjonsfeil. Måten vi har gjort det på sørger for at resultatet ikke har mer enn to desimaler.
Billigere-enn-metoden sammenligner prisen på de to varene. Den returnerer true hvis denne varen er billigere enn den andre, false hvis ikke. Her kunne vi valgt en int-metode som returnerer et positivt eller negativt tall avhengig av sammenligningen, men navnet metoden skulle ha tilsier at den har et ja/nei-svar.
Skriv-metoden kaller klassens toString-metode og skriver resultatet til terminal. For å gjøre dette har vi også overstyrt («override») toString-metoden. Vår toString-metode returnerer varens informasjon i et menneskelig lesbart format.

Utskrift:

```
Vare 1: #1 Testvare 1 (kr 150,00)
Vare 2: #2 Testvare 2 (kr 220,00)
Moms på vare 1: 30.0
Moms på vare 2: 44.0
Vare 1 billigere: true
```