HØGSKOLEN I BERGEN

BERGEN UNIVERSITY COLLEGE

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse: Dataingeniør + Informasjonsteknologi / H2014 / 1Data og 1Informajonsteknologi

Dato: 19. desember 2014

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall sider (inkludert denne)/ Tal på sider (medrekna denne): 8

Antall vedlegg/ Tal på vedlegg: 1 (API dokumentasjon for HashMap)

Tillatte hjelpemidler/ Tekniske hjelpemiddel: Ingen

Fagansvarlig/ Fagansvarleg:
Sven-Olai Høyland, Lars Michael Kristensen

Merknader/ Merknad: Ingen

Oppgave 1 (vekt 10%)

I hver deloppgave har du gitt et Java-program som kan kompileres og kjøres. Du skal svare hva som blir skrevet ut på skjermen når main blir kjørt. Merk at det kan være skrivesetninger i både main og metoden som blir kallet.

a)

```
public static void main(String[] args) {
   System.out.println(5 + 3 * 2);
   System.out.println(9 % 5);
   System.out.println(9 / 5);

   int i = 3;
   int j = 7;
   System.out.println( (i < 3) && (j >= 5) );
   System.out.println( (j != 3) || (i < j) );
}
```

b)

```
public static int b(int x, int y) {

   while (x != y) {
      System.out.println("x = " + x + ", y = " + y);
      if (x > y) {
         x = x - y;
      } else {
         y = y - x;
      }
   }

   return x;
}

public static void main(String[] args) {
   System.out.println(b(28, 12));
}
```

c)

```
public static int c(int x) {
   int i = 0;

   do {
      x = x / 10;
      i++;
      System.out.println("x = " + x);
   } while (x != 0);

   return i;
}

public static void main(String[] args) {
   System.out.println(c(32812));
}
```

d) Merk at det er to metoder med navnet `f`.

```
public static double f(int a, double b){
   return a - b;
}

public static double f(double a, int b){
   return a + b;
}

public static void main(String[] args) {
   System.out.println(f(4.0, 7));
}
```

e)

```
public static void e() {
   String[] sTab = new String[4];
   sTab[0] = "a";
   sTab[1] = "ab";
   sTab[2] = "abc";

   int totalLengde = 0;
   try {
      for (int i = 0; i < sTab.length; i++) {
         totalLengde += sTab[i].length();
         System.out.println("Lengde så langt: " + totalLengde);
      }
      System.out.println("Total lengde av strengene: " + totalLengde);
   } catch (ArithmeticException e) {
      System.out.println("Unntak nr 1 kasta.");
   } catch (NullPointerException e) {
      System.out.println("Unntak nr 2 kasta.");
   } catch (Exception e) {
      System.out.println("Unntak nr 3 kasta.");
   }
}

public static void main(String[] args) {
   e();
}
```

Oppgave 2 (vekt 20%)

Vi skal lage noen metoder for finne ledige bord på en restaurant.

Klassen `Bord` er gitt. Denne klassen har to objektvariabler, `antall` som gir oss hvor mange plasser totalt det er ved bordet og `ledig` som er **sann** (true) om bordet er ledig og **usann** (false) ellers. Klassen har get/set (hent/sett) metoder for objektvariablene. Disse kan brukes (og skal ikke lages).

Klassen `BordOversikt` inneholder en tabell (array) med pekere til Bord-objekt. Tabellen er full (alle posisjonene er i bruk). Deler av klassen er vist under. Vi skal lage tre objektmetoder (instansmetoder) i denne klassen.

```
public class BordOversikt {

      private Bord[] bt;
      …
      //a
      public int antallLedige() {…}

      //b
      public int finnFørsteLedige(int antall) {…}

      //c
      public int passerBest(int antall) {…}
}
```

a) Lag metoden `antallLedige()` som returnerer tallet på ledige bord.

b) Lag metoden `finnFørsteLedige(int antall)` som returnerer nummer (plass i tabellen) på første bordet som er ledig og er stort nok. Stort nok betyr at tallet på plasser ved bordet må være minst like stort som parameteren `antall`.

c) Lag metoden `passerBest(int antall)` som returnerer nummeret på minste bordet som er stort nok. Om det er flere bord av samme størrelse som passer best skal metoden returnere et av dem.

Eksempel: Om du søker etter et bord med plass til 4, så vil et bord med 6 plasser passe bedre enn et med 8 plasser (men et bord med plass til to er for lite).

Oppgave 3 (vekt 15%)

Vi har en fil med ord og ønsker å lage en frekvenstabell (tabell over hvor mange ganger hvert ord forekommer i filen). Vi har gitt en metode `lesDataFraFil()` for å lese ord fra filen inn i en tabell av strenger. For å lage frekvenstabellen og legge til rette for søking, bruker vi en HashMap.

Eksempel:

Dersom tabellen av strenger er: `{"er", "det", "er", "alle", "det", "det"}`, så blir frekvenstabellen (ikke tegnet som HashMap):

"er"	2
"det"	3
"alle"	1

Du har gitt starten av programmet (du trenger ikke gjenta dette i svaret). Du finner deler av API-dokumentasjonen for `HashMap` som vedlegg.

```
public static void main(String[] args) {

    HashMap<String, Integer> ft = new HashMap<String, Integer>();

    // Etter denne setningen er utført, ligg orda i tabellen data. Alle
    // posisjonene er i bruk
    String[] data = lesDataFraFil();

    // a
    // Lag frekvenstabellen

    // b
    // Vise hvordan du søker etter et ord i frekvenstabellen

    // c
    // Skriv ut frekvenstabellen

}
```

a) Lag frekvenstabellen ved hjelp av den deklarerte HashMap'en. Du går gjennom tabellen `data` (med strenger). Første gang du finner ordet setter du det inn i tabellen med frekvens 1. Om du har funnet ordet før, øker du frekvensen med 1.

b) Les inn et ord fra brukeren og skriv ut hvor mange ganger det finnes i frekvenstabellen (svaret kan være 0).

c) Skriv ut frekvenstabellen. Det er tilstrekkelig å få listet alle ord med tilhørende frekvens (trenger ikke sorteres på noen måte).

Oppgave 4 (vekt 35%)

I denne oppgaven skal vi skrive Java kode for klasser til et blogg-system på nettet. Dersom typen på argumenter til metoder eller typen på verdiene de returnerer ikke er gitt, må du bestemme typene selv.

a) Definer en klasse `Innlegg` med objektvariaber `bruker` (streng), `dato` (streng), og antall `likes` (heltall). Videre skal du lage get- og set- metoder for de tre objektvariablene. De tre objektvariablene skal ikke være synlige utenfor klassen og klassen skal ha en konstruktør `Innlegg(String bruker, String dato)` som setter `bruker` og `dato` for de to objektvariablene og setter antall `likes` til 0.

b) Definer en metode `skrivUt()` som skriver ut verdier for de tre objektvariablene `bruker`, `dato` og antall `likes` på skjermen.

c) Implementer en metode `atLike` som øker tallet på likes med en.

d) Definer to subklasser for klassen `Innlegg` med navnene `Bilde` og `Tekst`. `Bilde` skal ha en objektvariabel `url` (String) som gir en URL til der vi finner bildet. `Tekst` skal ha en objektvariabel `tekst` som er teksten i et blogg-innlegg.

e) Implementer konstruktører i de to klassene `Bilde` og `Tekst` som initialiserer alle objektvariablene (også de som er arvet fra superklassen) ut fra verdier som er gitt som argument til konstruktøren.

f) Implementer en metode `skrivUt()` i hver av subklassene `Bilde` og `Tekst` som skriver verdien av alle objektvariablene ut på skjermen (inkludert objekvariablene som er arvet fra superklassen).

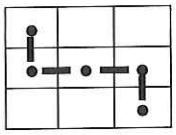
g) Skriv en main-metode som tillater brukaren å opprette et blogg-innlegg (bilde eller tekst) dvs. oppretter et `Bilde` eller et `Tekst` objekt ut fra input gitt av brukeren. Til slutt skal `skrivUt` metoden brukes for å skrive informasjonen om innlegget ut på skjermen.

Oppgave 5 (vekt 20%)

På datamaskiner med touch-skjerm som mobiltelefoner og nettbrett blir det ofte brukt et login-mønster på for eksempel 3x3 felt for å låse opp skjermen.

Et login-mønster kan representes ved bruk av en to-dimensjonal tabell av sannheitsverdier der `true` i en celle betyr at feltet er en del av mønsteret mens `false` betyr at cellen ikke er en del av mønsteret.

Eksempel: Følgende login-mønster med 3x3 celler



blir da representert som:

```
public class Login {

      private boolean[][] monster = { { true,  false, false },
                                      { true,  true,  true },
                                      { false, false, true } };
```

Du skal lage følgende objekt- / instansmetoder i klassen `Login`:

a) Skriv en metode `boolean erMed(int r, int k)` som returnerer `true` dersom cellen på rad r (heltall mellom 0 og 2) og kolonne k (heltall mellom 0 og 2) er med i login-mønsteret.

b) Skriv en metode `void skrivUt()` som skriver ut login-mønsteret på skjermen der tegnet '*' blir brukt for celler som er med i mønsteret og mellomrom (blank) blir brukt for de andre cellene. For eksempel skal utskrift av mønsteret gitt ovenfor skrives ut som:

```
*
***
  *
```

c) Skriv en metode `boolean sjekkMonster(boolean[][] mnstr)` som returnerer `true` dersom mønsteret `mnstr` gitt som parameter er det samme mønsteret som objektvariablen `monster` og `false` ellers. Du kan anta at `mnstr` er en to-dimensjonal tabell med 3 rader og 3 kolonner.

Lykke til!

Vedlegg: API-dokumentasjon for Class HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map
V - the type of mapped values

Method Summary

Modifier and Type	Method and Description
Void	clear() Removes all of the mappings from this map.
Boolean	containsKey(Object key) Returns true if this map contains a mapping for the specified key.
Boolean	containsValue(Object value) Returns true if this map maps one or more keys to the specified value.
V	get(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
Boolean	isEmpty() Returns true if this map contains no key-value mappings.
Set<K>	keySet() Returns a Set view of the keys contained in this map.
V	put(K key, V value) Associates the specified value with the specified key in this map.
V	remove(Object key) Removes the mapping for the specified key from this map if present.
boolean	remove(Object key, Object value) Removes the entry for the specified key only if it is currently mapped to the specified value.
V	replace(K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value.
boolean	replace(K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value.
int	size() Returns the number of key-value mappings in this map.
Collection<V>	values() Returns a Collection view of the values contained in this map.