BOKMÅL/NYNORSK

Høgskolen i Sørøst-Norge

EKSAMENSFORSIDE

Skriftlig (digital) eksamen med tilsyn

Emnekode: **6108 + 6108N**	Emnenavn: **Programmering i Java**	
Dato: **19.05.2017**	Tid fra / til: **10:00-14:00**	Ant. timer: **4**
Ansv. faglærer: **Roy M. Istad**		
Campus: **Bø**	Fakultet: **Handelshøyskolen**	
Antall oppgaver: **4**	Antall vedlegg: **1**	Ant. sider inkl. forside og vedlegg: **10**
Tillatte hjelpemidler (jfr. emnebeskrivelse): **Alt trykt og skrevet materiale.**		
Opplysninger om vedlegg: **Ett ark merket «Min-klasse». Dette er en samling med klassemetoder/verktøymetoder som studentene kan referere til i sine løsninger uten å skrive programkoden i disse metodene på nytt.**		
Merknader: **Ingen**		

Kryss av for type eksamenspapir	Ruter ☐		Linjer ☐

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG

BOKMÅL

Oppgave 1 * * * FLERVALG * * * Vekting: 20%

Angi spørsmålsnummer sammen med ditt svaralternativ, f.eks. slik: **1 a)**

Flervalgsdelen består av 10 spørsmål, der hvert spørsmål kun har ett riktig svaralternativ. Det er anledning til å gardere (dvs. angi flere svaralternativ). Riktig svaralternativ gir 3 poeng, (hvert) feil svaralternativ gir -1 poeng, mens et ubesvart spørsmål gir 0 poeng.

NB! Du kan forutsette at nødvendige import-setninger er på plass i alle kodeutsnittene.

1. Hva er verdien i variabelen `x` etter denne tilordningen dersom verdien i `a` er `5`?

```
int x = a + a;
```

a) 5
b) 6
c) 10
d) 55

2. Hva er verdien i variabelen `y` etter denne tilordningen dersom verdien i `b` er 10?

```
int y = 1 + b%20;
```

a) 0
b) 1
c) 1.5
d) 11

3. Hva er verdien i variabelen `z` dersom innholdet i `int`-variabelen `c` er lik 2?

```
String z = "c" + c;
```

a) Tegnsekvensen `cc`
b) Tegnsekvensen `c2`
c) Tegnsekvensen `22`
d) Tegnsekvensen `c+2`

4. Hva er verdien i variabelen `w` rett etter at **`if`**-setningen er utført?

```
double w = 1 + 2*2/5;
if ( w == 1 )
  w += 1;
else
  w = -1;
```

a) -1.0
b) 0.8
c) 2.0
d) 2.8

5. Hva blir skrevet i konsollet av denne sekvensen av programsetninger?

```
int sum = t[0];
for (int i=1; i<t.length; i++)    // Tabellen t har int-verdier
  sum += t[i] - t[i-1];           // i samtlige komponenter
out.println(sum);
```

a) Gjennomsnittet av alle tallene i `t`-tabellen
b) Differansen mellom alle partall og oddetall i `t`-tabellen
c) Det siste tallet i `t`-tabellen
d) Summen av alle tallene i `t`-tabellen

6. Hva blir skrevet i konsollet av denne sekvensen av programsetninger?

```
String txt = "5,2,7,0";
String[] d = txt.split(",");
out.println( d[2] + d[0] );
```

a) Ingen verdi pga. syntaksfeil
b) 7
c) 20
d) 75

7. Hva blir returverdien fra metodekallet `met('4')`? Metoden er gitt ved:

```
private int met(int n) {
  return ( 48 <= n && n <= 57 );
}
```

a) `true`
b) 52
c) `false`
d) Ingen verdi pga. syntaksfeil

8. Hvor mange *ganger* blir setningen i denne løkke-kroppen utført?

```
int p = 5;
do {
  p = p/2;
} while (p > 1);
```

a) 2
b) 3
c) Ukjent antall pga. evig løkke
d) Ingen utførelser pga. syntaksfeil

9. Hvilket alternativ vil du si omtaler modifikatoren **`final`** mest korrekt?
   a) Variabler beskyttes mot verdiendring ved å sette **`final`** i deklarasjonen.
   b) set- og get-metoder må brukes for å få tilgang til objektvariabler deklarert som **`final`**.
   c) Konstanter deklareres med **`final`** kun når datatypen er **`void`**.
   d) Hjelpemetoder med **`final`** på formelle parametre, må ha **`return`**-setning.

10. Hvordan vil du fullføre setningen: «*En hjelpemetode …* »
    a) kan returnere flere verdier i ulike datatyper.
    b) må være deklarert som **`void`** dersom den har flere enn én **`return`**-setning.
    c) er typisk deklarert som **`private`**.
    d) brukes ofte i stedet for konstruktør (ved overlasting/overloading).

*** * * SLUTT PÅ FLERVALG * * ***

Oppgave 2 — Vekting: 20%

2 a)
Skriv et komplett Java-program for å få tegnet et siffer-kvadrat i konsollet. Antall linjer (og kolonner) skal være en innlest verdi i området 2 – 9, som lagres i variabelen **`dim`** (dimensjon). Hver linje i kvadratet skal skrives med ett fast siffer lik hvert linjenummer, opp til det antallet linjer som **`dim`** tilsier.

Eksempel på tre ulike utskrifter: Én med **`dim`** lik 2, én med **`dim`** lik 3, og så én med **`dim`** lik 5.

```
1 1
2 2
```

```
1 1 1
2 2 2
3 3 3
```

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

2 b)
Skriv *kun* de Java-setningene som er nødvendige for å ta en serie med positive heltall, f.eks.

`heltall:` 2 2 2 2 2 0 0 8 8 2 17 17 17 17 17 17 8 8 8

og komprimere dem til en tekststreng. Tallene er lagret i en tabell kalt **`heltall`**, som ikke har noen tomme plasser (tabellen er full). Fra tabellen skal det bygges opp en ny tekststreng kalt **`komprimert`**, som sier hvilke positive heltall som inngår og hvor lange sekvenser de inngår i. Dette skal gjøres i den samme rekkefølgen som heltallene står i tabellen.

Dersom tallene gitt ovenfor er lagret i tabellen, så skal følgende tekststreng være resultatet:

`komprimert:` "2x5 0x2 8x2 2x1 17x6 8x3"

Det er brukt ett mellomrom (ett blankt tegn) for å skille de ulike elementene i den komprimerte tallserien. **NB!** Du skal altså ikke skrive et komplett program.

Oppgave 3 — Vekting: 30%

På sekvensielle tekstfiler har en skole lagret resultat fra gruppearbeid i noen fag. De ønsker et program som kan systematisere resultatene litt. Programmet skal gå gjennom en slik fil og for hver linje hente gruppenummer (første opplysning på linja), liste opp fornavn på medlemmene i gruppa, og beregne summen av poengene som hver av dem har oppnådd i gruppearbeidet.

Et eksempel, med kun noen av gruppene i denne klassen vist i detalj, er gitt på følgende figur, der de ulike dataelementene er adskilt med tegnsekvensen " – ":

klasse2a.txt

```
4 – Arne – 2 – Ida – 3 – Anne – 3 – Lars – 1
3 – Egil – 3 – Jon – 3 – Line – 5
7 – Knut – 2 – Aud – 2 – Kari – 5
:
:
2 – Lisa – 4 – Are – 3 – Mads – 2 – Mona – 1
```

Resultatet av systematiseringen skrives på en ny fil, med "res" foran opprinnelig filnavn, linje for linje i den rekkefølge de står på den opprinnelige datafila.

(Oppgaven fortsetter på neste side)

Dersom programmet behandler akkurat den fila som er vist på figuren foran, så vil følgende fil bli resultatet:

resklasse2a.txt

```
Gruppe 4 (Arne, Ida, Anne, Lars): 9 poeng
Gruppe 3 (Egil, Jon, Line): 11 poeng
Gruppe 7 (Knut, Aud, Kari): 9 poeng
:
:
Gruppe 2 (Lisa, Are, Mads, Mona): 10 poeng
```

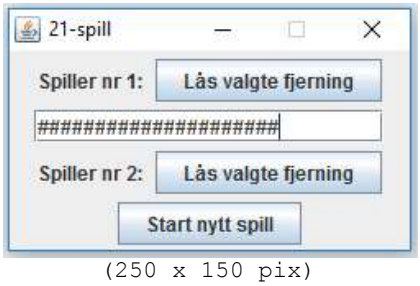
Skriv et komplett Java-program som denne skolen kan bruke til å systematisere opplysningene fra det nevnte gruppearbeidet. Programmet skal først be om og lese inn navnet på den aktuelle datafila, systematisere de gitte opplysningene og deretter skrive dem på en ny fil kalt "res" foran det opprinnelige filnavnet. Det kan forutsettes at alle aktuelle filer ligger i samme mappe som kjørende program.

Oppgave 4 — Vekting: 30%

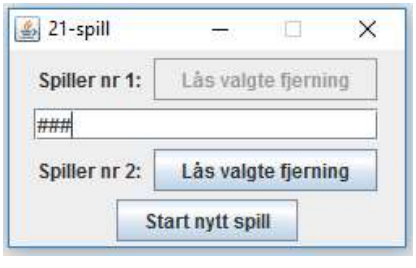
Tjueett, eller 21, er et strategispill der to deltagere, hver sin gang, fjerner enten én, to eller tre fyrstikker fra en haug med 21 fyrstikker (derav navnet). Den som trekker siste fyrstikk fra haugen vinner spillet. De to deltagerne trekker lodd (kaster mynt) om hvem som skal begynne spillet, dvs. være først til å fjerne fyrstikker fra haugen.

Skriv et GUI-program som simulerer *Tjueett*-spillet ved å bruke en sekvens av grindtegn, eller *hashtag* #, i stedet for haugen med fyrstikker. Spillet starter med 21 slike tegn i et tekstfelt (lengde 19). Første deltager sletter 1, 2 eller 3 grindtegn fra dette tekstfeltet og klikker på sin knapp merket med «Lås valgte fjerning». Da blir spillerens *knapp* deaktivert ved metodekallet `knapp.setEditable(false)`. Så er det den andre deltageren sin tur til å gjøre tilsvarende sletting og låse inn sitt valg via klikk på sin knapp. Da deaktiveres denne knappen, mens den første deltagerens knapp reaktiveres ved metodekallet `knapp.setEditable(true)`.

Slik veksler spillet frem og tilbake mellom de to, helt til alle grindtegnene er fjernet og spillet er avsluttet. Da vises det en melding i tekstfeltet om hvem som vant, dvs. hvem som fjernet siste grindtegn. NB! Programmet skal ikke tilby trekning av spill-rekkefølgen for deltagerne, det skal de avgjøre seg imellom utenom spillet (ved f.eks. myntkast).



(250 x 150 pix)



På figuren ovenfor er det vist startvindu for programmet på figuren til venstre, og der ser vi at begge knappene er klikkbare. På figuren til høyre er det slik at nr 2 kommer til å vinne spillet (ved å fjerne de tre siste grindtegnene) siden det er hans tur og han ennå ikke har fjernet noen grindtegn. Vi ser at nr 2 sin knapp er aktiv, mens knappen for deltager nr 1 er deaktivert etter at han gjorde sin siste fjerning av grindtegn.

— SLUTT —

NYNORSK

Oppgåve 1 * * * FLEIRVAL * * * Vekting: 20%

Angi spørsmålsnummer saman med ditt svaralternativ, f.eks. slik: **1 a)**

Fleirvalsdelen består av 10 spørsmål, der kvart spørsmål kun har eitt rett svaralternativ. Det er anledning til å gardere (dvs. angi fleire svaralternativ). Rett svaralternativ gir 3 poeng, (kvart) feil svaralternativ gir -1 poeng, mens eit ubesvart spørsmål gir 0 poeng.

NB! Du kan gå ut frå at nødvendige import-setningar er på plass i alle kodeutsnitta.

1. Kva er verdien i variabelen `x` etter denne tilordninga dersom verdien i `a` er `5`?

```
int x = a + a;
```

a) 5
b) 6
c) 10
d) 55

2. Kva er verdien i variabelen `y` etter denne tilordninga dersom verdien i `b` er 10?

```
int y = 1 + b%20;
```

a) 0
b) 1
c) 1.5
d) 11

3. Kva er verdien i variabelen `z` dersom innhaldet i `int`-variabelen `c` er lik 2?

```
String z = "c" + c;
```

a) Teiknsekvensen `cc`
b) Teiknsekvensen `c2`
c) Teiknsekvensen `22`
d) Teiknsekvensen `c+2`

4. Kva er verdien i variabelen `w` rett etter at **`if`**-setninga er utført?

```
double w = 1 + 2*2/5;
if ( w == 1 )
  w += 1;
else
  w = -1;
```

a) -1.0
b) 0.8
c) 2.0
d) 2.8

5. Kva blir skrevet i konsollet av denne sekvensen av programsetningar?

```
int sum = t[0];
for (int i=1; i<t.length; i++)    // Tabellen t har int-verdiar
  sum += t[i] - t[i-1];           // i samtlege komponentar
out.println(sum);
```

a) Gjennomsnittet av alle tala i `t`-tabellen
b) Differansen mellom alle partal og oddetal i `t`-tabellen
c) Det siste talet i `t`-tabellen
d) Summen av alle tala i `t`-tabellen

6. Kva blir skrevet i konsollet av denne sekvensen av programsetningar?

```
String txt = "5,2,7,0";
String[] d = txt.split(",");
out.println( d[2] + d[0] );
```

a) Ingen verdi pga. syntaksfeil
b) 7
c) 20
d) 75

7. Kva blir returverdien frå metodekallet `met('4')`? Metoden er gitt ved:

```
private int met(int n) {
  return ( 48 <= n && n <= 57 );
}
```

a) `true`
b) 52
c) `false`
d) Ingen verdi pga. syntaksfeil

8. Kor mange *ganger* blir setninga i denne løkke-kroppen utført?

```
int p = 5;
do {
  p = p/2;
} while (p > 1);
```

a) 2
b) 3
c) Ukjent antal pga. evig løkke
d) Inga utføring pga. syntaksfeil

9. Kva for eit alternativ vil du seie omtalar modifikatoren **`final`** mest korrekt?
   a) Variabler beskyttast mot verdiendring ved å sette **`final`** i deklarasjonen.
   b) set- og get-metodar må brukast for å få tilgang til objektvariablar deklarert som **`final`**.
   c) Konstantar deklarerast med **`final`** kun når datatypen er **`void`**.
   d) Hjelpemetodar med **`final`** på formelle parametrar, må ha **`return`**-setning.

10. Korleis vil du fullføre setninga: «*Ein hjelpemetode …* »
    a) kan returnere fleire verdiar i ulike datatypar.
    b) må være deklarert som **`void`** dersom han har meir enn ei **`return`**-setning.
    c) er typisk deklarert som **`private`**.
    d) brukast ofte i staden for konstruktør (ved overlasting/overloading).

*** * * SLUTT PÅ FLEIRVAL * * ***

Oppgåve 2 — Vekting: 20%

2 a)
Skriv eit komplett Java-program for å få teikna eit siffer-kvadrat i konsollet. Antal liner (og kolonner) skal vere ein innlest verdi i området 2 – 9, som lagrast i variabelen **`dim`** (dimensjon). Kvar line i kvadratet skal skrivast med eitt fast siffer lik kvart linenummer, opp til det antalet liner som **`dim`** tilseier.

Eksempel på tre ulike utskrifter: Ei med **`dim`** lik 2, ei med **`dim`** lik 3, og så ei med **`dim`** lik 5.

```
1 1
2 2
```

```
1 1 1
2 2 2
3 3 3
```

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

2 b)
Skriv *kun* dei Java-setningane som er nødvendige for å ta ein serie med positive heiltal, f.eks.

`heiltal:` 2 2 2 2 2 0 0 8 8 2 17 17 17 17 17 17 8 8 8

og komprimere han til ein tekststreng. Tala er lagra i ein tabell kalt **`heiltal`**, som ikkje har nokon tomme plassar (tabellen er full). Frå tabellen skal det byggast opp ein ny tekststreng kalt **`komprimert`**, som seier kva for positive heiltal som inngår og kor lange sekvensar dei inngår i. Dette skal gjerast i den same rekkefølgja som heiltala står i tabellen.

Dersom tala gitt ovanfor er lagra i tabellen, så skal følgjande tekststreng vere resultatet:

`komprimert:` "2x5 0x2 8x2 2x1 17x6 8x3"

Det er brukt eitt mellomrom (eitt blankt teikn) for å skilje dei ulike elementa i den komprimerte talserien. **NB!** Du skal altså ikkje skrive eit komplett program.

Oppgåve 3 — Vekting: 30%

På sekvensielle tekstfiler har ein skole lagra resultat frå gruppearbeid i nokre fag. Dei ønskjer eit program som kan systematisere resultata litt. Programmet skal gå gjennom ei slik fil og for kvar line hente gruppenummer (første opplysning på lina), liste opp fornavn på medlemmane i gruppa, og berekne summen av poenga som kvar av dei har oppnådd i gruppearbeidet.

Eit eksempel, med kun nokre av gruppene i denne klassen vist i detalj, er gitt på følgjande figur, der dei ulike dataelementa er adskilt med teiknsekvensen " – ":

klasse2a.txt

```
4 - Arne - 2 - Ida - 3 - Anne - 3 - Lars - 1
3 - Egil - 3 - Jon - 3 - Line - 5
7 - Knut - 2 - Aud - 2 - Kari - 5
:
:
2 - Lisa - 4 - Are - 3 - Mads - 2 - Mona - 1
```

Resultatet av systematiseringa skrivast på ei ny fil, med "res" foran opprinneleg filnamn, line for line i den rekkefølgja dei står på den opprinnelege datafila.

(Oppgåva held fram på neste side)

Dersom programmet behandlar akkurat den fila som er vist på figuren foran, så vil følgjande fil bli resultatet:

resklasse2a.txt

```
Gruppe 4 (Arne, Ida, Anne, Lars): 9 poeng
Gruppe 3 (Egil, Jon, Line): 11 poeng
Gruppe 7 (Knut, Aud, Kari): 9 poeng
:
:
Gruppe 2 (Lisa, Are, Mads, Mona): 10 poeng
```

Skriv eit komplett Java-program som denne skolen kan bruke til å systematisere opplysningane frå det nemnde gruppearbeidet. Programmet skal først be om og lese inn namnet på aktuell datafil, systematisere dei gitte opplysningane og deretter skrive dei på ei ny fil kalt "res" foran det opprinnelege filnamnet. Det kan forutsettast at alle aktuelle filer ligg i same mappe som køyrande program.

Oppgåve 4 — Vekting: 30%

Tjueeitt, eller 21, er eit strategispel der to deltakarar, kvar sin gang, fjerner enten ei, to eller tre fyrstikker frå ein haug med 21 fyrstikker (derav namnet). Den som trekker siste fyrstikk frå haugen vinn spelet. Dei to deltakarane trekker lodd (kastar mynt) om kven som skal begynne spelet, dvs. vere først til å fjerne fyrstikker frå haugen.

Skriv eit GUI-program som simulerer *Tjueeitt*-spelet ved å bruke ein sekvens av grindteikn, eller *hashtag* #, i staden for haugen med fyrstikker. Spelet starter med 21 slike teikn i eit tekstfelt (lengde 19). Første deltakar sletter 1, 2 eller 3 grindteikn frå tekstfeltet og klikkar på sin knapp merka med «Lås valde fjerning». Da blir spelaren sin *knapp* deaktivert ved metodekallet `knapp.setEditable(false)`. Så er det den andre deltakaren sin tur til å gjere tilsvarande sletting og låse inn sitt val via klikk på sin knapp. Da deaktiverast denne knappen, mens den første deltakaren sin knapp reaktiverast ved metodekallet `knapp.setEditable(true)`.

Slik vekslar spelet fram og tilbake mellom dei to, heilt til alle grindteikna er fjerna og spelet blir avslutta. Da visast det ei melding i tekstfeltet om kven som vann, dvs. kven som fjerna siste grindteikn. NB! Programmet skal ikkje tilby trekking av spel-rekkefølgja for deltakarane, det skal dei avgjere seg imellom utanom spelet (ved f.eks. myntkast).

(250 x 150 pix)

På figuren ovanfor er det vist startvindu for programmet på figuren til venstre, og der ser vi at begge knappane er klikkbare. På figuren til høgre er det slik at nr 2 kjem til å vinne spelet (ved å fjerne dei tre siste grindteikna) siden det er han sin tur og han ennå ikkje har fjerna noko grindteikn. Vi ser at nr 2 sin knapp er aktiv, mens knappen for deltakar nr 1 er deaktivert etter at han gjorde sin siste fjerning av grindteikn.

— SLUTT —

```
/*****************************************************************
 * Min: Verktøyklasse - dvs. samling av klassemetoder. Det er lov å
 * kalle på disse metodene fra program der det måtte være aktuelt.
 *****************************************************************
 */
import static javax.swing.JOptionPane.*;
import static java.lang.Integer.*;
import static java.lang.System.*;
import static java.lang.Math.*;

public class Min {

  // Metoden skriver en sekvens av tegn i en String
  public static String skrivTegn(char t, int antall) {
    String ut="";
    for (int i=1; i<=antall; i++)
      ut += t;
    return ut;
  }

  // Metoden avgjør om et tegn er en (engelsk) bokstav
  public static boolean erBokstav(char tegn) {
    return ('A'<=tegn && tegn<='Z') || ('a'<=tegn && tegn<='z');
  }

  public static char stor(char tegn) {
    if ('a'<= tegn && tegn<= 'z')
      return (char)(tegn - 32);
    return tegn; // Slår inn om tegn ikke er liten bokstav
  }

  // Metoden avrunder et tall til én desimal
  public static double avrund1(double tall) {
    double eps = 0.5;
    if ( tall < 0 ) eps = -0.5;
    return (int)(tall*10 + eps)/10.0;
  }

  // Metoden avrunder et tall til to desimaler
  public static double avrund2(double tall) {
    double eps = 0.5;
    if ( tall < 0 ) eps = -0.5;
    return (int)(tall*100 + eps)/100.0;
  }

  // Metoden leser inn et heltall i området (min-max)
  public static int lesHeltall(int min, int max) {
    int antall=0;
    do {
      String innTekst = showInputDialog("Gi heltall ("+min+"-"+max+"): ");
      antall = parseInt(innTekst);
      if ( antall < min || antall > max )
        showMessageDialog(null,"Ulovlig verdi!");
    } while (antall < min || antall > max);
    return antall;
  }

  // Metoden trekker et tilfeldig heltall i området min - max
  public static int trekkTall(int min, int max) {
    return min + (int)( random()*(max-min+1) );
  }

}
```