Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / H2017 /
1Data og 1Informajonsteknologi

Dato: 11. desember 2017

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: 2
(API dokumentasjon for Scanner og String)

Tillatte hjelpemidler/ Tekniske hjelpemiddel: Ingen

Fagansvarlig/ Fagansvarleg:
Sven-Olai Høyland (472 59 543), Lars Michael Kristensen (938 66 491)

Språk: Bokmål (nynorsk finnes i andre del av oppgavesettet)

Merknader/ Merknad: Ingen

Oppgave 1 (vekt 15%)

a) Hva blir skrevet ut når metodekallet `ma()` blir utført?

```
public static void ma() {

	int x = 2;
	int y = 4;

	double d = 2.0;
	char c = 'd';
	boolean b = false;

	System.out.println(d);

	System.out.println("x : " + x);

	System.out.println(x + x * y);

	System.out.println(x == 7 || b);

	System.out.println(!(x >= 7) && c == 'e');

	System.out.println(x * y * (-x));

	System.out.println(2 / 4);
}
```

b) Hva blir skrevet ut når metodekallet `mb()` blir utført?

```
public static void mb() {

	int[] tab = { 4, 1, -1, -7 };
	int t1 = -2;
	int t2 = 2;

	for (int i = 0; i < tab.length; i++) {
		int v = tab[i];

		if (v >= t1 && v <= t2) {
			System.out.println(v);
		}
	}

}
```

c) Hva blir skrevet ut når metodekallet `mc()` blir utført?

```
public static void mc() {

      int[] tab = { 1, 2, 3 };

      int i = 0;
      while (i < tab.length) {
            mc1(tab[i]);
            i++;
      }
}

public static void mc1(int x) {

      for (int i = 0; i < x; i++) {
            System.out.print("+");
      }

      System.out.println();
}
```

d) Metoden nedenfor bruker metoden `mc1` fra delspørsmål c). To av linjene i koden for metoden `md` vil gi en feilmelding fra kompilatoren. Hvilke linjer? Grunngi svaret.

```
public static int md() {

      boolean b = mc1(7);

      int y = 6;

      if (b) {
            int x = y + 1;
      }

      return x;
 }
```

Oppgave 2 (vekt 20%)

På flyplasser blir informasjon om flyavganger og status vist på skjermer. Vi skal lage noen klasser som kan brukes til å behandle og vise informasjon om flyavganger.

a) Lag klassen `Flyavgang`. Klassen skal inneholde:

- Objektvariabler som bare skal være synlige innenfor klassen: `nummer` (heltall), `destinasjon` (streng), `tid` (streng), og `status` (tegn).
- Konstruktør med parametre der vi kan gi verdi til alle objektvariablene.

Objektvariabelen `status` skal brukes slik at 'o' betyr "on-time", 'g' betyr "go-to-gate", 'b' betyr "boarding" og 'c' betyr "gate-closed". Tid blir representert som en streng på formen `tt:mm`, der `tt` angir time og `mm` angir minutt, eksempelvis `"23:03"` .

Klassen har også get()- og set()-metoder, men disse skal **ikke** lages. De kan brukes nedenfor ved behov.

b) Lag en metode `String toString()` som returnerer en streng på formen:

```
"541 Bergen 23:05 on-time\n"
```

Metoden skal se på tegn lagret i status og legge inn "on-time", "go-to-gate", "boarding" og "gate-closed" i strengen avhengig av verdien på status. Hvis status ikke inneholder tegnet 'o', 'g', 'b' eller 'c', skal metoden legge til "-" på slutten av strengen.

Vi ønsker å kunne behandle en samling av flyavganger slik vi kan vise informasjon om aktuelle flyavganger på en skjerm.

c) Lag en klasse `Flyavganger` som inneholder:

- en objektvariabel `tab` som er en peker til en referansetabell som inneholder pekere (objektreferanser) til objekter av klassen `Flyavgang` fra a). Objektvariabelen skal ikke være synlig utenfor klassen.
- Konstruktør med en parameter der vi kan gi størrelsen på `tab`. Konstruktøren skal opprette en tabell med korrekt størrelse gitt via parameteren og sette objektvariabelen `tab` til å peke på tabellen.

I denne varianten av samlinger skal **ikke** klassen ha noen objektvariabel som angir antall elementer som er lagret i samlingen og flyavgang-objektene skal ikke lagres fortløpende fra posisjon 0 og utover. Et element / en inngang i tabellen som inneholder en null-peker, skal vi forstå som at inngangen er ledig.

Eksempel: i en tabell med lengde fem som inneholder to flyavganger kan for eksempel posisjon 0 i tabellen peke på den første flyavgangen og posisjon 2 peke på den andre flyavgangen. Da inneholder tabellen `null` i posisjon 1, 3 og 4, og posisjon 1 er den første ledige plass/inngang i tabellen.

I de neste delspørsmålene vil vi legge til (implementere/lage) metoder i klassen `Flyavganger`.

d) Lag metoden `public boolean erNull(int p)` som returnerer `true` om inngangen på posisjon `p` i tabellen er `null` (en null-peker) og `false` ellers.

e) Lag metoden `public int antall()` som finner og returnerer antall flyavganger som er lagret i samlingen dvs. antall innganger i tabellen som ikke er `null`.

f) Lag metoden `public void visinfo()` som bruker `toString`-metoden fra klassen `Flyavgang` til å skrive ut informasjon om de flyavgangene som er lagret i tabellen.

Eksempel: for en tabell med to flyavganger skal utskrift på skjermen bli:

```
541 Bergen 23:05 on-time
543 Oslo 23:30 go-to-gate
```

g) Lag metoden `public boolean setStatus(int nummer, char status)` som setter status for flyavgangen lik det tegnet som er gitt ved parameteren `status` for flyavgangen gitt ved parameteren `nummer`. Metoden skal returnere `true` om flyavgangen angitt med `nummer` finnes i samlingen og `false` ellers.

h) Lag metoden `public boolean settinn(Flyavgang flyavgang)` som setter inn en ny flyavgang på første ledige plass i tabellen dvs. første posisjon som har en null-peker. Metoden skal returnere `false` om der ikke er plass i tabellen og `true` om flyavgangen ble satt inn. Du kan anta at flyavgangen som skal settes inn ikke finnes i tabellen fra før.

i) Lag metoden `public boolean slett(int nummer)` som sletter flyavgangen gitt ved parameteren `nummer` fra tabellen om flyavgangen finnes. Metoden skal returnere `true` om flyavgangen ble slettet og `false` ellers.

Oppgave 3 (vekt 20%)

Merk at API-dokumentasjon for en del metoder i klassene `Scanner` og `String` er lagt ved til slutt i oppgaveteksten.

Java-koden nedenfor inneholder deler av en metode som gitt et `filnavn` skal lese innholdet av filen linje for linje.

```
public static void les(String filnavn) throws FileNotFoundException
{

      File fil = new File(filnavn);
      Scanner sc = new Scanner(fil);

      // Her skal du skrive noe kode i deloppgave a) og b)

      leser.close();
}
```

a) Gjør ferdig implementasjonen av `les` metoden (markert ved kommentaren `//` ovenfor) slik at metoden leser innholdet av filen linje for linje ved bruk av Scanner-objektet `sc` og skriver linjen ut på skjermen.
Hint: se på metodene `hasNext` og `next` i Scanner-klassen.

b) Modifiser koden fra a) slik at det bare er linjer som ikke inneholder del-strengen `"//"` som blir skrevet ut på skjermen. Metoden kan da for eksempel brukes til å fjerne en-linjes kommentarer fra en java-fil.

c) Lag metoden `public static boolean lesCatch(String filnavn)` som kaller metoden `les` fra b) (for å lese og skrive ut filen) men som i tillegg fanger opp et eventuelt `FileNotFoundException` unntak som kan kastes av `les` metoden. Metoden skal returnere `true` om filen ble lest inn og skrevet ut og `false` om der ble signalert (reist/kastet) et `FileNotFoundException` unntak.

d) Lag metoden `public static void lesForsok(int forsok)` som gir brukeren et antall forsøk (gitt ved parameteren `forsok`) for å angi korrekt navn på filen som skal leses inn og skrives ut. Metoden skal bruke `lesCatch`–metoden fra c) til hvert forsøk på å lese filen. Hvis et forsøk på å lese filen blir mislykket, skal brukeren ha en melding "Feil" som blir vist ved å bruke `showMessageDialog(null, "Feil")` og brukeren får mulighet for å angi et nytt filnavn ved å bruke metoden `showInputDialog("Angi filnavn")` som returnerer strengen som brukeren skriver inn.

Oppgave 4 (vekt 25%)

En friidrettsklubb ønsker at medlemmene skal kunne konkurrere mot hverandre uavhengig av øvelse (for eksempel 100 m, spyd og høyde) og kjønn. For å få dette til, vil klubben gi poeng for oppnådde resultat etter regler som vi kommer tilbake til.

a) Definer en abstrakt klasse `Resultat`. Klassen skal inneholde:

   - objektvariablene `navn` (streng) og `kjønn` (tegn, 'K' for kvinne og 'M' for mann). Objektvariablene skal ikke være synlige utenfor klassen.
   - en konstruktør som gir verdi til begge objektvariablene.
   - `get/set`-metoder for objektvariablene.

b) Definer en abstrakt metode `int poeng()` i klassen `Resultat`.

c) Definer to subklasser `Spyd` og `HundreMeter`. `Spyd` har en objektvariabel `lengde` (desimaltall/flyttall). HundreMeter har en objektvariabel `tid` (desimaltall). Om du ønsker, kan du bruke forkortelsen `HM` for `HundreMeter`.

d) Implementer konstruktør i de to klassene `Spyd` og `HundreMeter` som intialiserer (gir startverdi) til alle objektvariablene (også de som er arvet fra superklassen) ut fra verdier som er gitt som argument til konstruktøren.

e) Implementer metoden `poeng()` i hver av subklassene `Spyd` og `HundreMeter`. Reglene (litt forenklet i forhold til de som er i bruk) er som følger:

   For klassen `Spyd`, vil et kast på 77.20 gi 1000 poeng for en mann. En kvinne må kaste 60.00 m for å få 1000 poeng. For hver meter man kaster lengre/kortere, vil man vinne/tape 15 poeng (gjelder for begge kjønn). **Eksempel**: En mann som kaster 78.20, vil få 1015 poeng. Poengene blir rundet ned til nærmeste heltall.

   For `HundreMeter`, vil en tid på 10.40 sekunder gi 1000 poeng for en mann. En kvinne må springe på 11.6 sekunder for å få 1000 poeng. For hvert sekund man er raskere/senere, vil man vinne/tape 240 poeng (gjelder begge kjønn). **Eksempel**: En kvinne som springer på 11.7, vil få 976 poeng. Poengene blir rundet ned til nærmeste heltall.

f) Lag en main-metode, der du oppretter en tabell med plass til 10 resultater. Sett inn et spydkast på plass 0 og en hundremeter på plass 1 i tabellen. Vi tenker oss at resten av tabellen blir fylt. Finn det beste resultatet i tabellen ved å bruke `poeng`-metoden og skriv det ut på skjermen. Du kan anta at alle klassene ovenfor har definert `toString()`-metoden.

Oppgave 5 (vekt 20%)

Vi kan se på en matrise som en todimensjonal tabell av tall. I denne oppgaven har vi heltall.
Eksempel:

1	2	3
4	-5	6
-7	8	0

Nedenfor har vi gitt første linjen av noen metoder vi skal lage:

```
public static boolean erPositiv(int[][] mat, int r, int k) {…}
public static boolean erPositivRekke(int[][] mat, int r) {…}
public static int antallNuller(int[][] mat) {…}
public static boolean erKvadratisk(int[][] mat) {…}
public static int spor(int[][] mat) {…}
```

a) Lag metoden `erPositiv(int[][] mat, int r, int k)` som returnerer `true` dersom tallet i rekke `r` og kolonne `k` i matrisen `mat` er større enn 0.

b) Lag metoden `erPositivRekke(int[][] mat, int r)` som returnerer true dersom **alle** tallene i rekke `r` i matrisen `mat` er større enn 0.

c) Lag metoden `antallNuller(int[][] mat)` som returnerer antall nuller i matrisen `mat` gitt som parameter. I eksemplet ovenfor skal metoden returnere 1.

d) Lag metoden `erKvadratisk(int[][] mat)` som returnerer `true` dersom antall rekker er lik antall kolonner i matrisen `mat`. I eksemplet ovenfor skal metoden returnere `true` siden begge er 3.

e) Lag metoden `spor(int[][] mat)` som returnerer summen av elementene på diagonalen i matrisen `mat`. Du kan anta at matrisen er kvadratisk. I eksemplet ovenfor skal metoden returnere 1 -5 + 0 = -4.

Lykke til!

Vedlegg 1: Utvalgt API-dokumentasjon for Scanner-klassen

Scanner(**File** source)
Constructs a new Scanner that produces values scanned from the specified file.

void	close() Closes this scanner.
boolean	hasNext() Returns true if this scanner has another token in its input.
boolean	hasNextLine() Returns true if there is another line in the input of this scanner.
String	next() Finds and returns the next complete token from this scanner.
int	nextInt() Scans the next token of the input as an int.
String	nextLine() Advances this scanner past the current line and returns the input that was skipped.
Scanner	skip(Pattern pattern) Skips input that matches the specified pattern, ignoring delimiters.
Scanner	reset() Resets this scanner.

Vedlegg 2: Utvalgt API-dokumentasjon for String-klassen

char	charAt(int index) Returns the char value at the specified index.
String	concat(String str) Concatenates the specified string to the end of this string.
boolean	contains(String s) Returns true if and only if this string contains the specified sequence of char values.
boolean	equals(Object anObject) Compares this string to the specified object.
int	indexOf(int ch) Returns the index within this string of the first occurrence of the specified character.
int	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring or -1.
int	length() Returns the length of this string.
boolean	matches(String regex) Tells whether or not this string matches the given regular expression.
boolean	startsWith(String prefix, int toffset) Tests if the substring of this string beginning at the specified index starts with the specified prefix.
ing	substring(int beginIndex) Returns a string that is a substring of this string.
String	substring(int beginIndex, int endIndex) Returns a string that is a substring of this string.

Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / H2017 /
1Data og 1Informajonsteknologi

Dato: 11. desember 2017

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: 2
(API dokumentasjon for Scanner og String)

Tillatte hjelpemidler/ Tekniske hjelpemiddel: Ingen

Fagansvarlig/ Fagansvarleg:
Sven-Olai Høyland (472 59 543), Lars Michael Kristensen (938 66 491)

Språk: Nynorsk (bokmål finn du i første del av oppgavesettet)

Merknader/ Merknad: Ingen

Oppgåve 1 (vekt 15%)

a) Kva blir skrive ut når metodekallet `ma()` blir utført?

```
public static void ma() {

	int x = 2;
	int y = 4;

	double d = 2.0;
	char c = 'd';
	boolean b = false;

	System.out.println(d);

	System.out.println("x : " + x);

	System.out.println(x + x * y);

	System.out.println(x == 7 || b);

	System.out.println(!(x >= 7) && c == 'e');

	System.out.println(x * y * (-x));

	System.out.println(2 / 4);
}
```

b) Kva blir skrive ut når metodekallet `mb()` blir utført?

```
public static void mb() {

	int[] tab = { 4, 1, -1, -7 };
	int t1 = -2;
	int t2 = 2;

	for (int i = 0; i < tab.length; i++) {
		int v = tab[i];

		if (v >= t1 && v <= t2) {
			System.out.println(v);
		}
	}

}
```

c) Kva blir skrive ut når metodekallet `mc()` blir utført?

```
public static void mc() {

	int[] tab = { 1, 2, 3 };

	int i = 0;
	while (i < tab.length) {
		mc1(tab[i]);
		i++;
	}
}

public static void mc1(int x) {

	for (int i = 0; i < x; i++) {
		System.out.print("+");
	}

	System.out.println();
}
```

d) Metoden nedanfor brukar metoden `mc1` frå delspørsmål c). To av linjene i koden for metoden `md` vil gi ei feilmelding frå kompilatoren. Kva for linjer? Grunngi svaret.

```
public static int md() {

	boolean b = mc1(7);

	int y = 6;

	if (b) {
		int x = y + 1;
	}

	return x;
 }
```

Oppgåve 2 (vekt 20%)

På flyplassar blir informasjon om flyavgangar og status vist på skjermar. Vi skal lage nokre klassar som kan brukast til å behandle og vise informasjon om flyavgangar.

a) Lag klassen `Flyavgang`. Klassen skal innehalde:

- Objektvariablar som berre skal vere synlege innanfor klassen: `nummer` (heiltal), `destinasjon` (streng), `tid` (streng), og `status` (teikn).
- Konstruktør med parametrar der vi kan gi verdi til alle objektvariablane.

Objektvariabelen `status` skal brukast slik at 'o' betyr "on-time", 'g' betyr "go-to-gate", 'b' betyr "boarding" og 'c' betyr "gate-closed". Tid blir representert som ein streng på forma `tt:mm`, der `tt` angir time og `mm` angir minutt, eksempelvis `"23:03"` .

Klassen har også get()- og set()-metodar, men desse skal **ikkje** lagast. Dei kan brukast nedanfor ved behov.

b) Lag ein metode `String toString()` som returnerer ein streng på forma:

```
"541 Bergen 23:05 on-time\n"
```

Metoden skal sjå på teikn lagra i status og legge inn "on-time", "go-to-gate", "boarding" og "gate-closed" i strengen avhengig av verdien på status. Viss status ikkje inneheld teiknet 'o', 'g', 'b' eller 'c', skal metoden legge til "-" på slutten av strengen.

Vi ønskjer å kunne behandle ei samling av flyavgangar slik vi kan vise informasjon om aktuelle flyavgangar på ein skjerm.

c) Lag ein klasse `Flyavganger` som inneheld:

- ein objektvariabel `tab` som er ein peikar til ein referansetabell som inneheld peikarar (objektreferansar) til objekt av klassen `Flyavgang` frå a). Objektvariabelen skal ikkje vere synleg utanfor klassen.
- Konstruktør med ein parameter der vi kan gi storleiken på `tab`. Konstruktøren skal opprette ein tabell med korrekt storleik gitt via parameteren og sette objektvariabelen `tab` til å peike på tabellen.

I denne varianten av samlingar skal **ikkje** klassen ha nokon objektvariabel som gir talet på element som er lagra i samlinga og flyavgang-objekt skal ikkje lagrast fortløpande frå posisjon 0 og utover. Eit element / ein inngang i tabellen som inneheld ein null-peikar, skal vi forstå som at inngangen er ledig.

Eksempel: i ein tabell med lengde fem som inneheld to flyavgangar kan for eksempel posisjon 0 i tabellen peike på den første flyavgangen og posisjon 2 peike på den andre flyavgangen. Då inneheld tabellen `null` i posisjon 1, 3 og 4, og posisjon 1 er den første ledige plass/inngang i tabellen.

I dei neste delspørsmåla vil vi legge til (implementere/lage) metodar i klassen `Flyavganger`.

d) Lag metoden `public boolean erNull(int p)` som returnerer `true` om inngangen på posisjon `p` i tabellen er `null` (ein null-peikar) og `false` elles.

e) Lag metoden `public int antall()` som finn og returnerer antal flyavgangar som er lagra i samlinga dvs. antal inngangar i tabellen som ikkje er `null`.

f) Lag metoden `public void visinfo()` som brukar `toString`-metoden frå klassen `Flyavgang` til å skrive ut informasjon om dei flyavgangane som er lagra i tabellen.

   Eksempel: for ein tabell med to flyavgangar skal utskrift på skjermen bli:

```
541 Bergen 23:05 on-time
543 Oslo 23:30 go-to-gate
```

g) Lag metoden `public boolean setStatus(int nummer, char status)` som set status lik det teiknet som er gitt ved parameteren `status` for flyavgangen gitt ved parameteren `nummer`. Metoden skal returnere `true` om flyavgangen angitt med `nummer` finst i samlinga og `false` elles.

h) Lag metoden `public boolean settinn(Flyavgang flyavgang)` som set inn ein ny flyavgang på første ledige plass i tabellen dvs. første posisjon som har ein null-peikar. Metoden skal returnere `false` om der ikkje er plass i tabellen og `true` om flyavgangen blei satt inn. Du kan anta at flyavgangen som skal settast inn ikkje finst i tabellen frå før.

i) Lag metoden `public boolean slett(int nummer)` som slettar flyavgangen gitt ved parameteren `nummer` frå tabellen om flyavgangen finst. Metoden skal returnere `true` om flyavgangen blei sletta og `false` elles.

Oppgåve 3 (vekt 20%)

Merk at API-dokumentasjon for ein del metodar i klassene `Scanner` og `String` er lagt ved til slutt i oppgåveteksten.

Java-koden nedanfor innheld deler av ein metode som gitt eit `filnavn` skal lese innhaldet av filen linje for linje.

```
public static void les(String filnavn) throws FileNotFoundException
{

      File fil = new File(filnavn);
      Scanner sc = new Scanner(fil);

      // Her skal du skrive noe kode i deloppgave a) og b)

      leser.close();
}
```

a) Gjer ferdig implementasjonen av `les` metoden (markert ved kommentaren `//` ovanfor) slik at metoden les innhaldet av filen linje for linje ved bruk av Scanner-objektet `sc` og skriv linja ut på skjermen.
Hint: sjå på metodane `hasNext` og `next` i Scanner-klassen.

b) Modifiser koden frå a) slik at det berre er linjer som ikkje inneheld del-strengen `"//"` som blir skrive ut på skjermen. Metoden kan då for eksempel brukast til å fjerne ein-linjes kommentarar frå ein java-fil.

c) Lag metoden `public static boolean lesCatch(String filnavn)` som kallar metoden `les` frå b) (for å lese og skrive ut filen) men som i tillegg fangar opp eit eventuelt `FileNotFoundException` unntak som kan kastast av `les` metoden. Metoden skal returnere `true` om filen blei lest inn og skrive ut og `false` om der blei signalert (reist/kasta) eit `FileNotFoundException` unntak.

d) Lag metoden `public static void lesForsok(int forsok)` som gir brukaren eit antal forsøk (gitt ved parameteren `forsok`) for å angi korrekt namn på filen som skal lesast inn og skrivast ut. Metoden skal bruke `lesCatch`–metoden frå c) til kvart forsøk på å lese filen. Viss eit forsøk på å lese filen blir mislykka, skal brukaren ha ei melding "Feil" som blir vist ved å bruke `showMessageDialog(null, "Feil")` og brukaren får muligheit for å angi eit nytt filnamn ved å bruke metoden `showInputDialog("Angi filnavn")` som returnerer strengen som brukaren skriv inn.

Oppgåve 4 (vekt 25%)

Ein friidrettsklubb ønskjer at medlemmene skal kunne konkurrere mot kvarandre uavhengig av øving (for eksempel 100 m, spyd og høgde) og kjønn. For å få dette til, vil klubben gi poeng for oppnådde resultat etter reglar som vi kjem tilbake til.

a) Definer ein abstrakt klasse `Resultat`. Klassen skal innehalde:

   - objektvariablane `namn` (streng) og `kjønn` (teikn, 'K' for kvinne og 'M' for mann). Objektvariablane skal ikkje vere synlege utanfor klassen.
   - ein konstruktør som gir verdi til begge objektvariablane.
   - `get/set`-metodar for objektvariablane.

b) Definer ein abstrakt metode `int poeng()` i klassen `Resultat`.

c) Definer to subklassar `Spyd` og `HundreMeter`. `Spyd` har ein objektvariabel `lengde` (desimaltal/flyttal). HundreMeter har ein objektvariabel `tid` (desimaltal). Om du ønskjer, kan du bruke forkortinga `HM` for `HundreMeter`.

d) Implementer konstruktør i dei to klassane `Spyd` og `HundreMeter` som intialiserer (gir startverdi) til alle objektvariablane (også dei som er arva frå superklasen) ut frå verdiar som er gitt som argument til konstruktøren.

e) Implementer metoden `poeng()` i kvar av subklassane `Spyd` og `HundreMeter`. Reglane (litt forenkla i forhold til dei som er i bruk) er som følgjer:

   For klassen `Spyd`, vil eit kast på 77.20 gi 1000 poeng for ein mann. Ei kvinne må kaste 60.00 m for å få 1000 poeng. For kvar meter ein kastar lengre/kortare, vil ein vinne/tape 15 poeng (gjeld for begge kjønn). **Eksempel**: Ein mann som kastar 78.20, vil få 1015 poeng. Poenga blir runda ned til næraste heiltal.

   For `HundreMeter`, vil ei tid på 10.40 sekund gi 1000 poeng for ein mann. Ei kvinne må springe på 11.6 sekund for å få 1000 poeng. For kvart sekund ein er raskare/seinare, vil ein vinne/tape 240 poeng (gjeld begge kjønn). **Eksempel**: Ei kvinne som spring på 11.7, vil få 976 poeng. Poenga blir runda ned til næraste heiltal.

f) Lag ein main-metode, der du opprettar ein tabell med plass til 10 resultat. Set inn eit spydkast på plass 0 og ein hundremeter på plass 1 i tabellen. Vi tenker oss at resten av tabellen blir fylt. Finn det beste resultatet i tabellen ved å bruke `poeng`-metoden og skriv det ut på skjermen. Du kan anta at alle klassane ovanfor har definert `toString()`-metoden.

Oppgåve 5 (vekt 20%)

Vi kan sjå på ei matrise som ein todimensjonal tabell av tal. I denne oppgåva har vi heiltal.
Eksempel:

1	2	3
4	-5	6
-7	8	0

Nedanfor har vi gitt første linja av nokre metodar vi skal lage:

```
public static boolean erPositiv(int[][] mat, int r, int k) {…}
public static boolean erPositivRekke(int[][] mat, int r) {…}
public static int antallNuller(int[][] mat) {…}
public static boolean erKvadratisk(int[][] mat) {…}
public static int spor(int[][] mat) {…}
```

a) Lag metoden `erPositiv(int[][] mat, int r, int k)` som returnerer `true` dersom talet i rekke `r` og kolonne `k` i matrisa `mat` er større enn 0.

b) Lag metoden `erPositivRekke(int[][] mat, int r)` som returnerer true dersom **alle** tala i rekke `r` i matrisa `mat` er større enn 0.

c) Lag metoden `antallNuller(int[][] mat)` som returnerer antal nullar i matrisa `mat` gitt som parameter. I eksemplet ovanfor skal metoden returnere 1.

d) Lag metoden `erKvadratisk(int[][] mat)` som returnerer `true` dersom talet på rekker er lik talet på kolonner i matrisa `mat`. I eksemplet ovanfor skal metoden returnere `true` sidan begge er 3.

e) Lag metoden `spor(int[][] mat)` som returnerer summen av elementa på diagonalen i matrisa `mat`. Du kan anta at matrisa er kvadratisk. I eksemplet ovanfor skal metoden returnere 1 -5 + 0 = -4.

Lukke til!

Vedlegg 1: Utvalgt API-dokumentasjon for Scanner-klassen

Scanner(**File** source)
Constructs a new Scanner that produces values scanned from the specified file.

void	close() Closes this scanner.
boolean	hasNext() Returns true if this scanner has another token in its input.
boolean	hasNextLine() Returns true if there is another line in the input of this scanner.
String	next() Finds and returns the next complete token from this scanner.
int	nextInt() Scans the next token of the input as an int.
String	nextLine() Advances this scanner past the current line and returns the input that was skipped.
Scanner	skip(Pattern pattern) Skips input that matches the specified pattern, ignoring delimiters.
Scanner	reset() Resets this scanner.

Vedlegg 2: Utvalgt API-dokumentasjon for String-klassen

char	charAt(int index) Returns the char value at the specified index.
String	concat(String str) Concatenates the specified string to the end of this string.
boolean	contains(String s) Returns true if and only if this string contains the specified sequence of char values.
boolean	equals(Object anObject) Compares this string to the specified object.
int	indexOf(int ch) Returns the index within this string of the first occurrence of the specified character.
int	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring or -1.
int	length() Returns the length of this string.
boolean	matches(String regex) Tells whether or not this string matches the given regular expression.
boolean	startsWith(String prefix, int toffset) Tests if the substring of this string beginning at the specified index starts with the specified prefix.
ing	substring(int beginIndex) Returns a string that is a substring of this string.
String	substring(int beginIndex, int endIndex) Returns a string that is a substring of this string.