Høgskolen i Telemark
Fakultet for allmennvitenskapelige fag

EKSAMEN

5609 000
OBJEKTORIENTERT PROGRAMMERING

03.12.2012

Tid:	5 timer, 9.00 – 14.00
Målform:	Bokmål
Sidetall:	Forside + 5 sider
Hjelpemidler:	Alle trykte og skrevne - ikke elektroniske. NetBeans IDE og Java JDK med Java API dokumentasjon er elektronisk tilgjengelig lokalt på eksamens PC.
Merknader:	**Besvarelsen skal leveres som papirutskrift påført kandidatnummer i første linje på hver ark.** **Lever besvarelsen i oppgaverekkefølge og skriv oppgavenummer foran hver delbesvarelse. Du må selv gå gjennom og kontrollere utskriften, og er selv ansvarlig for at det som leveres inn er komplett. Tid til utskrift og kontroll regnes ikke inn i eksamenstiden.**
Vedlegg:	Ingen

Besvarelsen kan skrives i NetBeans eller et tekstbehandlingsprogram (Word, TextPad eller Notisblokk). Besvarelsen skal skrives ut på eksamensskriveren og leveres på papir. Eksamensvaktene vil hjelpe til. Figurer o.l. som du har tegnet for hånd legges ved besvarelsen ved innlevering.

Sensuren finner du på Studentweb.

Start på oppgavesettet

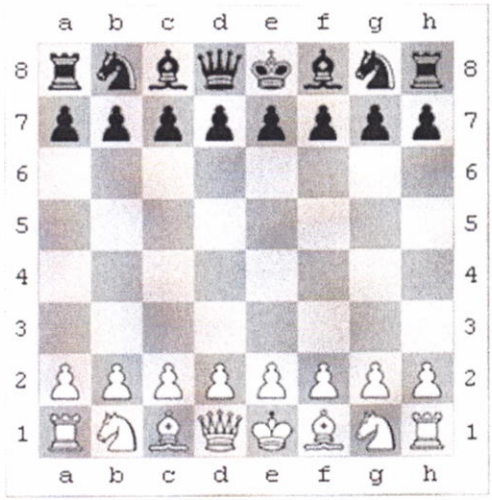
Oppgavene er uavhengige: Selv om det er en oppgave du ikke har løst, kan du løse neste som om den forrige var løst. Disponer tiden godt, slik at du får gjort noe på alle oppgavene. Om en oppgave er uklar så lag dine egne forutsetninger, og forklar disse.

Faktainformasjon

Du skal lage (deler av) et Java-program for sjakkspill. **Du trenger ikke å kunne noe om sjakk for å løse oppgaven, men her er litt faktainformasjon:**

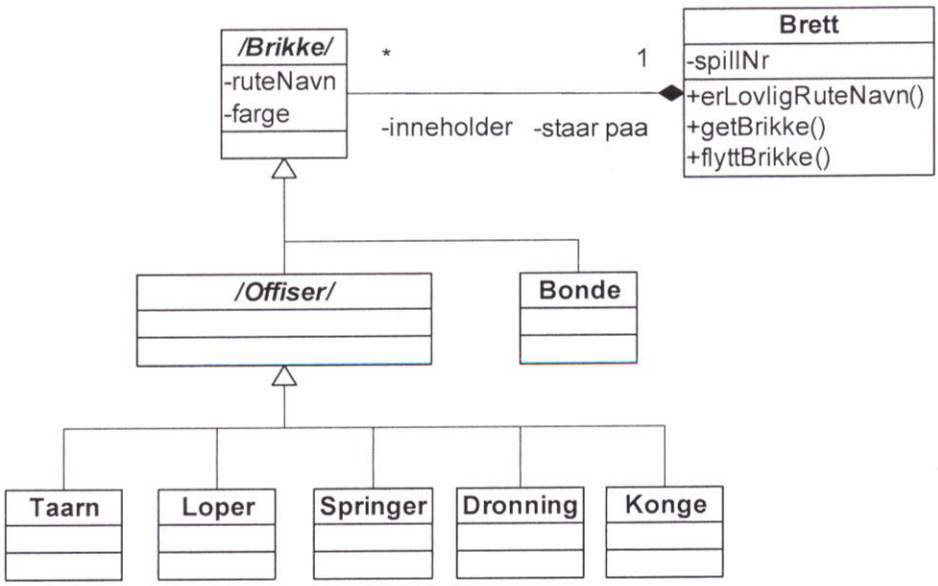
Sjakk spilles på et *brett* med 8 x 8 = 64 ruter. Radene er nummerert fra 1 til 8, og kolonnene har bokstaver fra a til h. To personer spiller mot hverandre. Hver spiller starter med 16 *brikker*, dvs. 32 brikker totalt. Den ene spilleren har **hvite** brikker og den andre har **svarte**.

Det er ulike typer brikker på brettet. I starten har hver spiller 8 *bønder*, 2 *tårn*, 2 *løpere*, 2 *springere* (hester), èn *dronning* og èn *konge*. I hver rute kan det maksimalt stå én brikke. Spillet går ut på å "slå ut" motparten sin konge.



Sjekkbrett med brikker i startposisjon

Det er laget et ufullstendig UML klassediagram over **modellklassene** i programmet. Klasser med klassenavn i */kursiv/* er abstrakte. Programmet skal bare behandle brikker som står på brettet. Brikker som er slått ut og fjernet fra brettet blir "kastet" og ikke tatt vare på.



Kodeskjelettet nedenfor beskriver deler av klassen **Brett**:

```
class Brett {
  public static final int BRETTSTORRELSE = 8;
  private int spillNr;
  private Brikke[][] brikkene;

  public static boolean erLovligRutenavn(String rutenavn)
  { skal kodes i oppgave 1a }
  public Brett(int nyttSpillNr)
  { skal kodes i oppgave 1d }
  public Brikke getBrikke(String rutenavn)
  { skal kodes i oppgave 1e }
  public Brikke flyttBrikke(String fraRute, String tilRute)
  { skal kodes i oppgave 1f }
}
```

I oppgave 1 a), c), d) og e) skal du programmere ferdig metodene i klassen. Du må selv vurdere om det er behov for flere "hjelpemetoder", variable eller konstanter i klassen.

Oppgave 1 Modellklasser (50 %)

1a)
Rutene på brettet er "nummerert" fra venstre til høyre (kolonnene) med små bokstaver a-h, og nedenfra og opp (radene) med sifrene 1-8. Hver rute har dermed et entydig *rutenavn* som består av én liten bokstav etterfulgt av ett siffer, f.eks. "b3", "f5".

Programmer metoden erLovligRutenavn() i klassen Brett. Metoden skal undersøke om rutenavnet er lovlig etter reglene over.

1b)
Programmer klassen Brikke i henhold til klassediagrammet på første side og beskrivelsen nedenfor. Begrunn kort de valgene og forutsetningene du gjør underveis. Obs! Du skal ikke programmere noen av subklassene til Brikke nå.

- Objektvariabelen **spillNr** identifiserer hvert spill/brett. Når et nytt brett opprettes får det et nytt spillnr.
- Programmer konstruktør og relevante get- og set-metoder slik at:
  - Alle brikker vet hvilken rute den står i og har et gyldig rutenavn.
  - Brikkenes farge er enten hvit eller svart.
  - Alle brikker alltid har en referanse til brettet de står på.
- Alle brikker skal ha en metode **erLovligTrekk()** som får inn et rutenavn, undersøker om denne brikken kan flytte seg dit, og returnerer svaret som true eller false. Tips: De ulike brikketypene har forskjellige regler for hvordan de kan flytte seg, men det er ikke er nødvendig å kjenne disse for å løse denne oppgaven.
- Alle brikker skal ha en metode **brikkenavn()** som returnerer en tekststreng med en bokstav som viser hva slags brikke den er, f.eks. "T" for tårn, "B" for bonde osv.
- Alle brikker skal ha en metode **flyttTil()** som får inn et rutenavn, undersøker om denne brikken kan flytte seg dit, flytter seg dit ved å endre brikken sitt rutenavn dersom trekket er lovlig, og returnerer true. Hvis ikke returnerer metoden false.

1c)
I denne oppgaven kan du anta at alle subklassene til Brikke er ferdig programmert. Brettet skal ha full oversikt over hvilke ruter brikkene står i. Til dette benyttes den todimensjonale tabellen **`brikkene`** i klassen **`Brett`**. I de rutene der det ikke står en brikke, peker referansene i matrisen på **`null`**.

Lag en konstruktør for klassen `Brett` som bl.a. oppretter tabellen, der første dimensjon i tabellen skal representere kolonner og den andre rader.

Konstruktøren i Brett skal også opprette de 32 brikkene og sette dem inn i tabellen på sine startposisjoner for et nytt spill. For å begrense arbeidet i denne oppgaven, skal du bare sette inn disse brikkene:

- De fire tårnene skal stå i hvert sitt hjørne av brettet, to hvite i rad 1 og to svarte i rad 8.
- I rad 2 og 7 skal det stå én bonde i hver rute. Rad 2 har bare hvite brikker og rad 7 bare svarte.

1d)
Programmer ferdig metoden getBrikke i klassen Brett. Metoden skal returnere en referanse til den brikken som står i ruten med det rutenavnet som sendes som parameter til metoden, eller **`null`** dersom det ikke er noen brikke der.

Tips: Kollonnebokstaver kan benyttes som kolonneindeks, f.eks. slik:

```
char kolonnebokstav;
int kollonneindeks = kolonnebokstav - 'a';
```

1e)
Programmer ferdig metoden `flyttBrikke` i klassen `Brett`.
Metoden skal forsøke å flytte brikken i ruten med navnet `fraRute` til ruten med navnet `tilRute`. Dersom trekket er gjennomførbart skal metoden returnere true, ellers returneres false.

Obs! Å flytte en brikke medfører både at brikkens rutenavn endres, og at brikken blir flyttet til en annen rute i brettets tabell. Hvis ruten som brikken flyttes til er "opptatt" av en annen brikke, blir den andre slått ut og forsvinner fra brettet.

1f) **(krevende)**
Programmèr klassene Offiser og Tårn.

- Det er ingen nye objektvariabler i klassen **Offiser**.
- Tårnet har lov til å flytte seg innenfor samme rad eller samme kolonne. Tårnet (og alle andre brikker) kan flytte til en rute der det står en annen brikke og "tar" da denne brikken, men tårnet kan ikke hoppe over noen brikker. (For de som kan sjakk: Vi ser bort fra det spesielle trekket "rokade" i denne oppgaven.)

Du skal ikke programmere de andre subklassene til Brikke.

Oppgave 2 (35 %)

I denne oppgaven skal du programmere et swing-GUI for sjakkprogrammet som ser slik ut:

Krav til funksjonalitet:

- Når programmet startes, skal det opprette et brett-objekt og lage et GUI som viser brettet med brikker som vist i figuren over.
- Brikkene vises på brettet med sitt brikkenavn (bokstav).
- Under brettet er det felt for å skrive inn neste trekk. Spilleren skriver rutenavn til én av brikkene i *Fra rute*, og navn på ruten som brikken skal flyttes til i *Til rute*.
- Programmet brukes av èn spiller for å spille mot maskinen. Spilleren flytter alltid de **sorte** brikkene. GUIet skal derfor sjekke at det står en sort brikke i *Fra rute*. (Du skal ikke endre metoden **erLovligTrekk** eller **flyttBrikke**. Disse skal også kunne flytte maskinens hvite brikker.)
- Du skal **ikke** programmere maskinens (mot-)trekk. ☺
- Når brukeren trykker knappen **Flytt**, skal brikken flyttes (hvis mulig) og hele brettet (alle brikkene) tegnes på nytt.
- Hvis brikken ikke er svart, ikke kan flyttes til oppgitt rute, eller det ikke står noen brikke i oppgitt fra-rute, skal brettet være uendret. Det skal da skrives en feilmelding: **Ugyldig trekk!** øverst i vinduet.
- Knappen **Lagre** skal kalle metoden **lagreBrett** i klassen **BrettLagrer** (Se oppgave 3).
- Knappen **Nytt spill** skal starte et nytt spill med et nytt brett-objekt i utgangsposisjon.

Tips:

- Sjakkbrettet og brikkene kan "tegnes" med draw-metoder i **Graphics** klassen (se Java APIet på eksamens-PCen).
- Hver gang brukeren flytter en brikke bør hele brettet tegnes på nytt.
- Rutene i sjakkbrettet kan tegnes med ulik gråfarge (Color.GRAY / Color.LIGHTGRAY).
- I oppgave 1 har vi bare fylt brettet med bønder og tårn, men lag GUIet slik at det virker uansett hvor mange brikker som står på brettet og hvilken posisjon de har.

Oppgave 3 Database (15 %)

Se figur av GUI i oppgave 2:
I denne oppgaven skal du lagre opplysninger om brikkene på brettet i en MySQL database.

Du skal her programmere metoden `lagreBrett` nedenfor. Denne kjøres med Lagre knappen i oppgave 2, men du kan løse denne oppgaven uten å ha løst oppgave 2.

Det er definert en MySQL database med en tabell som heter **Brikker** med følgende kolonner:

spillnr	INTEGER	
brikkenavn	CHAR(1)	
farge	CHAR(1)	("H" = hvit, "S" = svart)
rutenavn	CHAR(2)	

Det er laget et kodeskjelett til en ny klasse **BrettLagrer** som ser slik ut:

```
import java.sql.*;

public class BrettLagrer {
    private static final String dbURL =
            "jdbc:mysql://db-kurs.hit.no:3306/sjakk";
    private static final String brukernavn = "sjakkmester";
    private static final String passord = "passord";

    public BrettLagrer() { }

    public static boolean lagreBrett(Brett skalLagres)
    { skal kodes i denne oppgaven}
}
```

Du skal programmere ferdig metoden `lagreBrett` slik at:

- Metoden lagrer alle brikkene på brettet i databasetabellen **Brikker**. (Det kreves altså flere SQL setninger som må kjøres for å lagre hele brettet.)
- Det skal være mulig å ta pause flere ganger i samme spill, dvs. man må kunne lagre samme brett flere ganger. Eventuell gammel versjon av brettet skal da slettes fra databasen.
- Lag eventuelt dine egne hjelpemetoder der det er hensiktsmessig.

Slutt på oppgavesettet.