INF1000 - Obligatorisk innlevering 5

Temaer denne uka: *Klasser og objekter*.

Oppgave 5.1)

Tema: *Klasser og objekter*
Filnavn: `Oppgave51.java, Bil.java`

Oppgaven går ut på å modellere biler. Biler kan ikke kjøre dersom de ikke har nok bensin. Hver kilometer en bil kjøres, legges til den totale kilometerstanden.

a) Gitt følgende grensesnitt, fyll ut metodene og bruk fornuftige variabler i klassen for å få metodene til å returnere korrekte verdier.

```
class Bil {
    // Nok bensin? Kjør "km" kilometer.
    public void kjorTur(int km) { }

    // Fyll tanken, dersom det er plass til spesifisert mengde bensin.
    public void fyllTank(double liter) { }

    // Hent maksimal kjøredistanse, gitt bensinmengde på tanken.
    public double hentMaksDistanse() { }

    // Hent bilens totale kilometerstand.
    public int hentKilometerstand() { }

}
```

Vi antar her at alle biler kjøpes nye (kilometerstand 0), med full tank. Det er nødvendig å spesifisere tankstørrelse og bensinforbruk pr. kilometer ved opprettelse av en bil. Et tips kan være å lagre disse verdiene i `final`-variabler. Dersom noe skulle gå galt, for eksempel at du forsøker å kjøre lenger enn du har bensin til, skal metodene gi tilbakemelding om at dette ikke går til terminalen.

b) Lag klassen `Oppgave51` med en `main`-metode. Opprett en bil, og ta den på noen kjøreturer. Ta bilen på en tur som er lenger enn maksimal distanse gitt

bensinmengde, og sjekk at programmet ditt gir tilbakemelding om at du ikke har nok bensin til turen. Skriv til slutt ut bilens totale kilometerstand.

Synes du denne oppgaven var vanskelig? Se øvingsoppgaver 5.01, 5.02 og 5.03.

Oppgave 5.2)

Tema: *Teorioppgave*
Filnavn: `Teori.txt`

Gi korte svar på spørsmålene under:

a) Hva er innkapsling? Hvorfor er det nyttig?

b) Hva er grensesnittet til en klasse? Hvordan skiller det seg fra implementasjonen av en klasse?

c) Hva er en instansmetode og hvordan skiller den seg fra en statisk metode (slike metoder vi har jobbet med tidligere)?

Oppgave 5.3-5.4 (gir 2 poeng!)

Tema: *Objektorientering*
Filnavn: `Oppgave54.java, Isbod.java`

Det er sommer, det er varmt, og folket vil ha is! Du skal lage et enkelt system som skal holde styr på ansatte for sjefen i en isbod. Å være ansatt i isbod er meget sesongbetinget, og hvor mange ansatte som trengs varierer fra uke til uke. Vi antar her at de som har vært ansatt kortest (lavest ansiennitet), vil bli avskjediget først, dersom noen må gå.

a) Skriv en klasse `Isbod`. Lag en privat String-array med ti plasser i klassen. Denne skal inneholde navn på de ansatte. Lag også en heltallsvariabel for å lagre antall ansatte.

b) Skriv en metode `public void ansett(String navn)`, som registrerer en ny ansatt. Navnet på den nyansatte skal lagres på første ledige plass i arrayen (har man 5 ansatte, skal neste ansatte komme på plass 5 i arrayen). Husk at du må sjekke at det ikke alt er 10 ansatte – da kan du ikke ansette flere!

c) Skriv en metode `public void giSistemannSparken()`, som gir den som sist ble ansatt sparken. Programmet skal da gi en utskrift med navnet på den som ble ansatt, fjerne vedkomne fra arrayen. Husk å senke antallet ansatte med én.

d) Skriv en metode `public void printAlleAnsatte()`, som skriver ut alle ansatte i rekkefølge, der den første har lengst ansiennitet, og den siste har kortest. Til denne oppgaven skal du bruke en for-løkke.

e) Lag en klasse `Oppgave54`, og opprett en main-metode. Lag et Isbod-objekt. Ansett tre personer. Skriv ut alle ansatte, og sjekk at korrekte navn kommer ut. Spark så den siste, og skriv ut på nytt. Sjekk at utskrift nok en gang blir korrekt.

Du har nå implementert en konstruksjon som kalles en "stack". Dette er en datastruktur som brukes mye i "det virkelige programmeringsliv", og som du nok kommer til å jobbe mye med som programmerer!

Oppgave 5.5)

Tema: *Egen oppgave*
Filnavn: `MinOppgave5.java`

Lag din egen oppgave der du bruker klasser og objekter. Både oppgaveteksten og besvarelsen skal leveres inn.

Krav til innleveringen

1. Klassenavnet og filnavnet skal være identisk.
2. Klassenavn skal skrives med stor forbokstav.
3. Variabelnavn skal ha liten forbokstav.
4. Oppgaven må kunne kompilere og kjøre på IFI sine maskiner.
5. Kun .java-filen skal innleveres.
6. Ikke bruk æ, ø eller å i .java-filene(heller ikke som kommentarer eller utskrift).
7. Filene skal inneholde gode kommentarer som forklarer hva programmet gjør.
8. Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.
9. Metodenavn skal skrives med liten forbokstav.
10. Koden skal være riktig indendert. Er du usikker, se Appendix J i Big Java.
11. Hver klasse skal ligge i sin egen .java-fil.

Fremgangsmåte for innleveringer i INF1000

1. Lag en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   - Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   - Hvor lang tid (ca) brukte du på innleveringen?
   - Var det noen oppgaver du ikke fikk til? Hvis ja:
     - Hvilke(n) oppgave er det som ikke fungerer i innleveringen?
     - Hvorfor tror du at oppgaven ikke fungerer?
     - Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?
2. Logg inn på Devilry.
3. Lever de 8 .java-filene, samt teori.txt, README.txt og filene du bruker i den egne oppgaven din i *samme innlevering*.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.

Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her og flere utfordringsoppgaver her.