Øvingsoppgaver – OOP (Objektorientert programmering)

# Oppgave 1

Du skal skrive en klasse ***Motorsykkel*** som skal modellere kjøringen av motorsykler. En motorsykkel har et merke, et registreringsnummer og en kilometerstand som viser hvor langt den har kjørt.

1. Implementer klassen ***Motorsykkel***med en konstruktør som setter egenskapene som klassen trenger.
2. Implementer en metode ***kjør(km)***som øker kilometerstanden med det gitte antall kilometer.
3. Implementer metoden ***hentKilometerstand()*** som skal returnere motorsykkelens totale kilometerstand.
4. Implementer metoden ***skrivUt()*** som skal logge merke, registreringsnummer og kilometerstand til konsollen.
5. Lag et UML-diagram av klassen.
6. Test at metodene i klassen fungere slik dem skal ved å opprette et motorsykkelobjekt med et merke, registreringsnummer og kilometerstand du selv velger. Opprett to til slike objekter. Øk kilometerstanden til den siste motorsykkelen du opprettet og sjekk at metoden er riktig ved å kalle på *hentKilometerstand().*

# Oppgave 2

I denne oppgaven skal du skrive en klasse ***Hund*** som har egenskapene vekt og alder som settes gjennom konstruktøren. Den skal også ha en egenskap som heter metthet som skal ha verdien 10.

1. Implementer klassen ***Hund***.
2. Implementer to metoder som returnerer henholdsvis alder og vekt.
3. Implementer metoden ***spring()***. Denne metoden skal redusere metthet med 1 og som i tillegg minsker vekten hvis mettheten er mindre enn 5.
4. Implementer metoden ***spis(verdi)***. Parameteren ***verdi***er et heltall og metoden skal øke mettheten med denne verdien. Vekten skal øke dersom mettheten er større enn 7.
5. Lag et UML-diagram av klassen.
6. Test klassen ved å opprette et hundeobjekt. Kall på metodene ***spis og spring*** minst 2 ganger hver og logg vekt til konsollen etter hvert kall for å sjekke at metodene gjør slik dem skal.

# Oppgave 3

I denne oppgaven skal du skrive en klasse ***Dato*** som skal representere en dato. Konstruktøren til klassen skal ha parameterne ***nyDag, nyMåned***og ***nyttÅr.*** Disse parameterne brukes til å gi verdier til samme egenskaper (dag, måned, år). Disse skal alle være heltall.

1. Implementer klassen ***Dato***
2. Utvid klassen med metoder som utfører følgende operasjoner på en dato. Velg parametere til hver metode og om den skal returnere en verdi etter hva du synes er hensiktsmessig.
3. Implementer metoden ***lesÅr*** som skal returnere årstallet for datoen.
4. Implementer en metode som lager og returnerer en streng av datoen. Strengen skal være slik at datoen er lett lesbar.
5. Implementer en metode som sjekker om datoen er en bestemt dag i måneden og som returnerer true eller false.
6. Implementer en metode som tar en ny dato og sjekker om den er før eller etter dette datoobjektet.\*
7. Implementer en metode som endrer datoen til neste dag.\*

Oppgave d) og e) er valgfrie ettersom de er noe utfordrende.

1. Lag et UML-diagram av klassen
2. Test klassen ved å opprette et datoobjekt der dagen er 15. Logg datoens år til terminalen. Logg datoen til terminalen som en lesbar streng. Hvis du gjorde d) og e) kan du teste disse metodene også

# Oppgave 4

Du skal lage en klasse ***Person*** med en konstruktør som tar imot et navn og en alder. Disse to egenskapene skal også opprettes og gis en verdi i konstruktøren. I tillegg skal klassen ha en egenskap ***hobbyer*** som er et tomt array. Klassen skal ha en metode ***leggTilHobby(hobby)*** som tar imot en streng som representerer en hobby og legge den til arrayet for hobbyer. Klassen skal også ha en metode ***skrivHobbyer()*** som skal logge alle hobbyene i listen til konsollen på hver sin linje. Klassen skal ha en tredje metode ***skrivUt()***som skal logge til konsollen navn og alder på personen i tillegg til hele listen med hobbyer .

1. Lag et UML diagram for klassen.
2. Implementer klassen.
3. Gjør nødvendige tester for å teste at alle metoder virker som de skal.

# Oppgave 5

I denne oppgaven skal du lage to klasser til et program for å finne og spille musikk. Klassene er beskrevet nedenfor.

Sang

Du skal skrive en klasse ***Sang*** med en konstruktør som har parametere for tittel og artist. Konstruktøren skal opprette disse samme egenskapene og gi dem parameterverdiene. Parameterne skal være av typen streng. F.eks en tittel kan være «Nothing Else Matters» og artist kan være «Metallica».

Klassen skal også tilby følgende metoder:

* **spill() –** Denne metoden «spiller av» sangen ved at den logger «Spiller <info om tittel og artist>» til konsollen.
* **sjekkArtist(navn) –** Parameteren **navn** er en streng som representerer en artist. Metoden skal returnere true hvis ett eller flere av navnene i strengen **navn** finnes i **artist**, og false hvis ikke.
* **sjekkTittel(tittel)** – Parameteren **tittel** er en streng som representerer en sangtittel. Metoden skal returnere true dersom parameteren **tittel** og **tittel** til sangobjektet er den samme, og false hvis ikke. Titlene skal regnes som like uavhengig av små og store bokstaver.
* **sjekkArtistOgTittel(artist, tittel)** – Metoden returnerer true dersom både artist og tittel er like ved ovenfornevnte regler, og false hvis ikke.
* **toString()** – Denne metoden skal returnere en streng der informasjon om sangen (artist og tittel) skrives ut på en lesbar, brukervennlig måte.

Spilleliste

Klassen ***Spilleliste***  skal ikke ha noen parametere i konstruktøren, men skal ha et navn og et array som skal inneholde Sang-objekter. Konstruktøren skal også legge til tre sanger i denne arrayen. Du velge selv hvilke sanger du vil opprette.

Klassen skal også tilby følgende metoder:

* **leggTilSang(sang)** – Parameteren **sang** er et Sang-objekt og skal legges til listen med sanger.
* **fjernSang(sang)** – Parameteren **sang** er et Sang-objekt. Metoden skal fjerne dette sangobjektet fra listen hvis den eksistere i listen.
* **spillSang(sang)** – Metoden skal spille av sangen oppgitt som parameter hvis den finnes i listen.
* **spillAlle()** – Metoden skal spille av alle sangene i spillelisten
* **finnSang(tittel)** – Metoden skal returnere første sang med tittel som matcher parameteren. Hvis tittelen ikke finnes i listen returneres **null**.
* **hentArtistUtvalg(artistnavn) –** Metoden skal returnere et array med sanger som har artisten fra parameteren som artist. Dette inkluderer også sanger der artisten ikke er alene som artist på sangen.

1. Lag et UML-diagram for begge klassene som viser relasjonen mellom dem
2. Implementer klassene
3. Gjør nødvendige tester for å sjekke at klassenes metoder fungerer slik dem skal.

# Oppgave 6

I denne oppgaven skal du skrive to klasser der den ene arver fra den andre. Klassen ***Bil***skal ha et merke og en verdi som settes gjennom konstruktøren. Klassen skal også ha en metode som heter **regnUtPris()** som tar bilens verdi og legger til 25% moms av bilens verdi. Metoden skal returnere prisen på bilen.

Klassen ***Elbil*** skal arve fra ***Bil***. I tillegg til å ha de samme egenskapene har elbilen en egenskap for rekkevidde som også settes gjennom kontruktøren. Rekkevidde er et heltall som representerer bilens rekkevidde i kilometer. Denne klassen skal også ha en metode **regnUtPris(),** men skal overskrives ved å bare returnere bilens verdi da elbiler er unntatt moms.

1. Lag et UML-diagram for klassene som viser forholdet mellom dem.
2. Implementer klassene.
3. Test klassene på en hensiktsmessig måte.