Plenum: Arv og polymorfi

Ahmed Waseem Saeed

Februar 2024

Introduksjon

Velkommen til ny plenumstime. Ukens tema er en videreføring av arv og subklasser, der vi skal se nærmere på konseptet polymorfi. Vi kommer til å benytte mekanismer fra forrige uke, samt introdusere konsepter som override, overload, og nøkkelordet super. Polymorfi gir oss muligheten til å bestemme semantikken til en metode basert på hvilken klasse som utfører metoden. Vi skal i dag modellere ulike varer. Vi skal skille mellom matvarer (godteri og frukt) og elektroniske varer (datamaskin). I tillegg skal vi implementere metoder som oppfører seg ulikt, avhengig av klassen de tilhører (eks. beregning av moms).

Klassehierarki

Tegn et klassehierarki med alle klassene i oppgaveteksten. *Husk å lese hele oppgaveteksten før du begynner.*

Vare

Klassen Vare er abstrakt, og har kun en instansvariabel pris (før mva), double, som settes i konstruktøren. I tillegg har alle varer en metode *beregnFullPris* som returnerer prisen inkludert merverdiavgift (mva). Mva-en for en uspesifisert vare kan settes til 0.25 (25 %).

ElVare

Den abstrakte klassen ElVare representerer elektroniske varer, og er en subklasse av Vare. Følgende instansvariabler må settes (og tas inn som parametre) i konstruktøren: batteritid (int, i antall timer), og aarSidenKjop (int, som tilsvarer antall år siden varen ble kjøpt).

Alle ElVare-objekter må kunne beregne den resterende batteritiden sin, gjennom metoden *beregnBatteritid*. For ElVarer så holder det å returnere variabelen

batteritid (siden vi ikke kan opprette objekter av ElVare, så er alle batterier fulladde). I tillegg skal du også overskrive *beregnFullPris* ved å bruke en mva på 0.3 (30 %).

Datamaskin

Klassen Datamaskin tar inn RAM (heltall), merke (streng), samt en batteristatus (double mellom 0 og 1 som angir prosent som er igjen), i tillegg til andre nødvendige variabler i foreldreklassen(e). Klassen er en subklasse av ElVare, og må kalle på super() for de gitte konstruktørene. Overskriv metoden *beregnBatteritid*, slik at metoden returnerer batteritid * batteristatus, istedenfor kun batteritiden.

Matvare

Matvare er en abstrakt klasse som er en subklasse av Vare. En Matvare tar inn en sunnhetsgrad (1 til og med 10) som spesifiserer hvor sunn en matvare er, samt et navn (streng). Hvis sunnhetsgraden er ugyldig, så skal programmet sette sunnhetsgraden automatisk til 1. Mva for Matvare settes til 0.14 (14 %). Skriv i tillegg metoden *erNokkelhullsmerket* som returnerer en Boolean. For Matvare kan den returnere True dersom sunnhetsgraden er over eller lik 6, ellers False.

Godteri

Klassen Godteri er en subklasse av Matvare og tar inn en phVerdi (int, 0 til og med 14) som angir surheten på godteriet, i tillegg til andre nødvendige parametre som trengs i de andre konstruktørene. Skriv i tillegg en til konstruktør som ikke tar en pH-verdi, men setter denne til -1 (tilsier uspesifisert pH) (kalles å overloade en metode). Mva for Godteri er 0.2 (20 %).

Metoden *erNokkelhullsmerket* skal kun returnere True her dersom sunnhetsgraden er over eller lik 9, samt phVerdien over 10.

Frukt

Klassen Frukt er en subklasse av Matvare, og har en mva på 0. Metoden *erNokkelhullsmerket* skal alltid returnere True her.

Testprogram

Skriv til slutt et program der du oppretter noen objekter av klassen Frukt, Datamaskin og Godteri, der du tester at funksjonaliteten ved å kalle på noen polymorfe metoder endrer seg slik den er tiltenkt.

Bonus: Handleliste.java

Opprett en klasse, Handlelste, som skal ha en main-metode der du legger inn legger inn ulike objekter som arver fra klassen Vare, inn i en liste. Til slutt skal programmet regne totalen for hele handlelisten ved å iterere gjennom lista, og skrive ut totalprisen på en ryddig måte.