Plenum: Arv og interface

Ahmed Waseem Saeed

Februar 2024

Introduksjon

Velkommen til ny plenumstime! Tema for denne uken er arv og interface, og med det så runder vi av rekken med arv. Ukens plenumstime kommer til å fokusere hovedsakelig på konseptet interface (norsk: grensesnitt), men vi kommer til å bygge på konsepter fra de foregående ukene. Denne uken skal vi modellere personer og ansatte, i tillegg til en bedrift. Legg merke til hvordan vi skal bruke interface for å spesifisere oppførselen som vi vil at en gitt klasse skal ha.

Klassehierarki

Tegn et klassehieraki som inkluderer alle klasser og interfacer nevnt i oppgaven (kan være at du må tegne to hierarkier). Som vanlig, les hele oppgaveteksten før du begynner.

Person

I motsetning til tidligere Person-klasser skrevet hittil i kurset, skal vi operette et interface først som spesifiserer hvilke egenskaper en Person-klasse bør ha. Skriv interfacet ErPerson som har følgende metoder som en del av signaturen:

```
String hentNavn();

int hentAlder();

void skrivPerson();
```

Skriv deretter klassen Person som implementerer dette interfacet. La interfacet guide hvordan du setter opp klassen, slik at du har det du trenger for å implementere metodene. Klassen skal dessuten være abstrakt.

Ansatt

Klassen Ansatt er en subklasse av Person, og implementerer et nytt interface, HarKontrakt, som spesifiserer detaljer i kontrakten til den ansatte. Følgende metoder er med i interfacet:

```
int hentLonn();

void lonnsOekning(int oekning);

boolean harHjemmekontor();
```

Igjen, likt som ved Person-klassen, skriv Ansatt-klassen i tråd med implementasjonskravene fra interfacet. I tillegg skal du overskrive metoden skrivPerson slik at den tar med relevant informasjon for et Ansatt-objekt.

Styremedlem

I vår lille verden anser vi ikke styremedlemmer som ansatte, men de implementerer fremdeles interfacet HarKontrakt. I tillegg arver klassen Styremedlem fra Person. Vi sier her at styremedlemmer ikke har hjemmekontor, og har i tillegg en bonus i tillegg til lønnen sin (tas inn som parameter i konstruktøren). Dessuten må ethvert styremedlem ha en rolle (String). Også denne klassen skal skrive over skrivPerson-metoden.

Bonus: Hvordan kunne vi implementert Styremedlem annerledes dersom vi hadde sagt at alle styremedlemmer må være ansatte?

Teste det vi har

Skriv et testprogram som oppretter et par Ansatt- og Styremedlemmer-objekter. Kall på skrivPerson() for objektene og verifiser at utskriften er slik du ønsket.

Bedrift

Vi ønsker nå opprette funksjonalitet for en bedrift. Klassen Bedrift har et navn, og år den ble grunnlagt. I tillegg implementerer klassen to grensesnitt: HarStyre og HarAnsatte. En bedrift må også ha opplysning om sin nåværende situasjon. Du kan enten la dette være en streng som heter status, eller opprette en enum (OBS: nytt) som har tre ulike statuser (BRA, DAARLIG, HELT_OK).

HarStyre ser slik ut:

```
ArrayList<Styremedlem> hentStyre();

// Tar en beslutning basert paa statusen til bedriften.
taBeslutning();
```

```
void leggTil(Styremedlem medlem);
```

Her står du med en del frihet til å spesifisere hva som skal skje i klassen ved kall på taBeslutning(), f.eks. kan du velge å printe en melding basert på statusen til bedriften.

HarAnsatte ser slik ut:

```
ArrayList<Ansatt> hentAnsatte();

Ansatt giSparken();

void ansett(Ansatt ny);
```

Uten å si mye om innmaten i klassen, så kan vi allerede se enkelte variabler og oppførsel som må være til stede for at implementasjonen til interfacene skal gi mening. Metoden giSparken skal gi sparken til den ansatte som har vært lengst i selskapet (hint: hvilken datastruktur har en egenskap der vi kan replikere en kø (først inn, først ut)). Metoden ansett() skal ansette en person ved å legge de til i datastrukturen som har oversikt over de ansatte i bedriften.

Skriv til slutt en metode skrivBedrift() som iterer alle ansatte og styremedlemmer i bedriften og skriver ut informasjon om dem.

Hovedprogram

Enten bygg videre på testprogrammet fra tidligere, eller opprett et nytt program som du kaller Hovedprogram. Opprett en Bedrift der du legger til ansatte og styremedlemmer. Skriv deretter ut alle i bedriften.

Egen Exception

Akkurat nå får vi en feil dersom vi prøver å sparke en Ansatt uten at vi har ansatte i bedriften. Skriv en egen Exception, som du kaller IngenAnsattException, som utvider RuntimeException. Denne skal kastes dersom man prøver å gi sparken i en bedrift uten ansatte. Du står fritt til å skrive hjelpemetoder dersom du ønsker det.