DAT102 – Våren 2024

Øvingsoppgaver uke2 (8.-12. januar)

Oppgave 1

Oppgaven prøver å illustrere en viktig forskjell mellom en «vanlig» for-løkke og en utvidet for-løkke.

Gitt klassene `Heltall` og `Main` nedenfor. Spørsmålene a) – e) er gitt i koden. For å løse oppgaven, gå fram på følgende måte.

1. Studer først koden nøye alene og svar på spørsmålene.
2. Diskuter svarene med en eller flere medstudenter og prøv å bli enige.
3. Kjør koden. Om noe blir feil, prøv å forstå hvorfor. Spør eventuelt om hjelp.

```
public class Heltall {
	private int tall;

	public Heltall(int n) {
		tall = n;
	}

	public void inkrementer() {
		tall++;
	}

	@Override
	public String toString() {
		return "" + tall;
	}
}
```

```
public class Main {

	static void skrivTab(int[] tab) {
		for (int i = 0; i < tab.length; i++) {
			System.out.print(tab[i] + " ");
		}

		System.out.println();
	}

	public static void main(String[] args) {

		int[] hTab = {1, 2, 3, 4};
		skrivTab(hTab);

		// a) Hva blir skrevet ut?
		for (int i = 0; i < hTab.length; i++) {
			hTab[i]++;
		}
		skrivTab(hTab);
```

```
            // b) Hva blir skrevet ut?
            for (int t : hTab) {
                  t++;
            }
            skrivTab(hTab);

            Heltall[] rTab = new Heltall[4];
            rTab[0] = new Heltall(1);
            rTab[1] = new Heltall(2);
            rTab[2] = new Heltall(3);
            rTab[3] = new Heltall(4);

            // c) Hvorfor fungerer ikke denne?
            // skrivTab(rTab);

            // d) Hva blir skrevet ut?
            for (Heltall t : rTab) {
                  System.out.print(t + " ");
            }
            System.out.println();

            // e) Hva blir skrevet ut?
            for (Heltall t : rTab) {
                  t.inkrementer();
            }

            for (Heltall t : rTab) {
                  System.out.print(t + " ");
            }
            System.out.println();
      }
}
```

Oppgave 2

Gitt Figur-grensesnittet under:

```
public interface Figur {
	double areal();
	void tegn();
}
```

Siden alle metoder i et grensesnitt er public er det vanlig å utelate public framfor double.

a) Lag en klasse Rektangel som implementer Figur-grensesnittet. Klassen skal ha medlemsvariablene høyde og bredde. Et rektangel med høyde 2 og bredde 3, skal tegnes slik:
***
***

b) Lag en klasse Trekant implementerer Figur-grensesnittet. Klassen representerer en rettvinklet trekant med den rette vinkelen nede til venstre . Klassen skal ha medlemsvariablen sideLengde. En trekant med sidelengde 3 tegnes slik:
*
**
***

c) Lag en Figur-liste og legg til noen rektangler og noen trekanter.

d) Beregn samlet areal for alle figurene i listen.

e) Tegn alle figurene under hverandre med en blank linje mellom.

Felles for oppgave 3 –5

Java har mange forhåndsdefinerte grensesnitt/kontrakter (interface). Dette gjelder blant annet for liste (eng. list), mengde (eng. set) og avbildning (eng. map). Videre finnes det flere forhåndsdefinerte klasser som implementerer disse grensesnittene. Vi skal vise hvordan man kan implementere disse (og andre) kontrakter i dette kurset. Disse oppgavene vil gi dere litt erfaring med å bruke forhåndsdefinerte klasser i Java. Dere må regne med å lese / lete i dokumentasjonen (som det er lenket til i oppgavene).

Oppgave 3

Grensesnitt List

Lenke: https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Generiske typer

Om vi bruker grensesnittreferanser (interface-referanser) i stedet for objektreferanser, så vil koden bli lettere å vedlikeholde. Dette gjelder også når vi deklarer formelle parametre eller returverdi for metoder. Om vi ønsker å skifte implementasjon, så er det tilstrekkelig å endre kall av konstruktøren. Grunnen til dette er at grensesnittreferanse kan referere til alle objekter av en klasse som implementerer grensesnittet.

I DAT100 så vi på generelle objektsamlinger og registrerte at om vi skal ha en samling med personer eller en samling av bøker, så er operasjonene de samme. Dermed kunne vi bruke Object som type. Det gir to problem. Vi må konvertere til rett type når vi tar ut et element frå samlingen. Det er også mulig å putte feil type objekt inn i samlingen. Java har mulighet å angi typeparametre til klasser. Da vil du slippe konverteringen og du vil få feil om du prøver å sette inn element av feil type. Eksempel:

```
List<String> liste = new ArrayList<>();
```

Det er anbefalt å la <> knyttet til konstruktøren vere tom slik som vist, men det er lov å skrive `<String>`. Nå vil du slippe å konvertere elementene til `String` når du tar ut siden du ikke får lov å sette inn andre objekt enn strenger. Legg også merke til at vi har brukt grensesnittreferanse (List) til objektet. Da vil det være enkelt å skifte til en annen listetype enn ArrayList.

Oppgave

a) Opprett en ArrayList som kan inneholde heltall. Legg til 5 elementer (heltall) der to er like. Se også hva som skjer om du prøver å legge til en streng. Skriv ut elementet på plass 4 og 5. Forklar hvorfor du får en feil.

b) Skriv ut antall elementer i listen ved bruk av en metode fra grensesnittet List. Legg til et nytt element på plass 2 (elementene er nummert fra 0). Skriv ut listen.

c) Sjekk om et element finnes i listen. Prøv både med et som finnes og et som ikke finnes.

d) Finn posisjon for første forekomst av elementene som er like. Finn deretter posisjon for siste forekomst.

e) Fjern alle elementer i listen og sjekk at den er tom.

Oppgave 4

Grensesnitt Set

Lenke: https://docs.oracle.com/javase/8/docs/api/java/util/Set.html

Svarer til mengder i matematikken. Merk at vi må lage `equals`-metoden for klasser som vi lager selv. Husk at når vi definerer `equals`-metoden for en klasse, skal vi alltid også definere metoden hashCode slik at metoden gir samme verdi for objekter som er like.

a) Lag en main-metode der du lager du oppretter to mengder som kan inneholde strenger. Legg til 3 – 5 strenger i hver av mengdene der minst en av strengene er felles. Skriv ut mengdene med en utvidet for-løkke. Du kan bruke klassen HashSet som implementerer Set-interfacet.

Når det gjelder operasjoner på mengder vil ofte operasjonene inneholder, snitt og union være nyttige.

b) Finn snittet av de to mengdene og skriv ut svaret. Ved å lese dokumentasjonen nøye, vil du finne en metode som passer. NB! Om du ønsker å beholde de originale mengdene, må du ta kopi av (minst) en av mengdene på forhånd. Dette kan du gjøre ved å lage en tom mengde og så bruke metoden `addAll()` med mengden du ønsker å kopiere som parameter.

c) Finn unionen av de to mengdene og skriv ut svaret. Se ellers spørsmålet over.

Oppgave 5

Grensesnitt Map

Lenke: https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Knytter sammen nøkkel og verdi. Merk at både nøkkel og verdi må være referansetyper. Det betyr at om en eller begge er primitive typer, må vi bruke tilhørende wrapper-klasse (int -> Integer, double -> Double, osv.). Nøklene må være unike som for eksempel personnummer eller studentnummer. Det kan bare være en verdi knyttet til en nøkkel, men flere nøkler kan være knyttet til samme verdi. Et eksempel på bruk er en telefonliste der vi antar navnene er unike: (Ole, 12345678), (Kari, 87654321), osv.

Oppgave

Vi har en liste av ord der hvert ord kan forekomme flere ganger. Vi ønsker å lage en frekvensoversikt (antall ganger hvert ord forekommer).

Eksempel: Dersom listen av ord er: {"er", "det", "alle", "er", "det", "det"}

så blir frekvensoversikten:

er: 2
det: 3
alle: 1

Vi skal bruke de forhåndsdefinerte klassene ArrayList og HashMap i Java for å løse oppgaven.

a) Lag først en liste der du for eksempel legger inn ordene i eksemplet ovenfor. Gå gjennom listen av ord og lag frekvensoversikten. Første gang du finner ordet, setter du ordet inn i HashMap'en med frekvens 1. Om ordet finnes fra før, øker du frekvensen med en.

b) Les inn et ord fra bruker og skriv ut tilhørende frekvens. Svaret kan være 0.

Oppgave 6

Gitt koden nedenfor. Du finner spørsmålene som kommentarer i koden.

```
// Hvor i programmet er x-en nedenfor gyldig?
static int f(int x) {
	x = 2 * x;
	return x;
}

static void g(double[] a) {
	for (int i = 0; i < a.length; i++) {
		a[i] = 2 * a[i];
	}

	// Ikke anbefalt, men lovlig kode
	a = new double[2];
	a[0] = 1.0;
	a[1] = 2.0;
}

public static void main(String[] args) {
	int[] htab = {2, 7};
	int a = 4;
	int b = f(a);
	// Hva er a og b nå?

	double[] tab = {2.0, 5.0};
	g(tab);
	// Hva er innholdet i tab nå?

	// Hvilke av disse setningene er lovlige?
	double c1 = f(3.0);
	double c2 = f(3);
	int c3 = f(3.0);
	int c4 = f(3);
}
```

Oppgave 7

a) Lag en klasse Student som har medlemsvariabler studentnummer (som er unikt), fornavn og etternavn og studieby/studiested. Lag en konstruktør som kan gi verdi til alle medlemsvariabler. Lag get- og set-metoder for alle medlemsvariabler. Disse kan genereres automatisk ved å velge «Source» på meny-linjen i Eclipse.

b) Klassen skal implementere grensesnittet (interfacet) Comparable slik at når du sammenligner to studenter så kommer den med minst studentnummer først.
https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

Ved å implementere dette grensesnittet, vil man kunne sortere en samling Student-objekter med de forhåndsdefinerte metoder i Arrays- eller Collections-klassen.

c) Overkjør toString()-metoden i klassen slik at den returnerer en streng med alle medlemsvariablene uten linjeskift.

d) Generer equals- og hachCode-metode automatisk. Velg «Source» på meny-linjen i Eclipse og kryss av for studentnummer. Husk at når du lager equals-metode, så skal du alltid også lage hashCode slik at like elementer får samme verdi på hashCode.

e) Lag en main-metode der du oppretter en liste der du bruker en av de forhåndsdefinerte listeklassene i Java (for eksempel ArrayList eller LinkedList). Lag 5 Student-objekter som du legger til listen slik at listen ikke er sortert med hensyn til studentnummer.

f) Collections-klassen har en del nyttige klassemetoder (static-metoder). Husk at klassemetoder kalles ved å skrive <navnPåKlassen>.<metodenavn>(<aktuelle parametre>);
https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

Sorter listen ved å bruke en av metodene i klassen. Skriv deretter listen ut ved å skrive System.out.println(<navnPåListe>); Dette fungerer siden ArrayList har definert toString()-metoden.

g) Vi ønsker at studentene blir sortert alfabetisk på etternavn, eventuelt fornavn om de har samme etternavn. Gjør nødvendige endringer i b). Senere vil dere lære andre måter å gjøre dette på.

Oppgave 8

a) Du skal lage en ny klasse NettStudent som arver fra student. I tillegg til medlemsvariabene som arves, skal klassen ha hjemsted.

b) Lag en get- og set-metoder og en konstruktør som kan gi verdi til alle medlemsvariablene.

c) Overkjør toString-metoden.

d) Lag en liste med 4-5 nettstudenter. Vi ønsker å skrive ut alle hjemsteder, men hvert hjemsted bare en gang. Dette kan gjøres på mange ulike måter, men ved å bruke en av de forhånddefinerte klassene i Java blir det lite kode.

e) Er det mulig å legge en NettStudent til en liste av Student-objekter? Diskuter gjerne med en medstudent før dere prøver.

f) Er det mulig å legge en Student til en liste av NettStudent-objekter? Diskuter gjerne med en medstudent før dere prøver.