NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

KONTINUASJONSEKSAMEN I FAG
TDT4100 Objekt-orientert programmering

Onsdag 17. august 2005
KL. 09.00 – 13.00

Faglig kontakt under eksamen:
Hallvard Trætteberg, tlf (735)93443 / 918 97263
Trond Aalberg, tlf (735)97952 / 976 31088

Tillatte hjelpemidler:

- Lewis & Loftus: Java Software Solutions (alle utgaver)
- Winder & Roberts: Developing Java Software
- Lervik & Havdal: Programmering i Java
- Mughal, Hamre & Rasmussen: Java som første programmeringsspråk
- Arnold, Gosling & Holms: The Java Programming Language
- Lervik & Havdal: Java the UML Way
- Liang: Introduction to Java programming
- Lewis & Loftus: Java Software Solutions
- Horton: Beginning Java 2 SDK 1.4 Edition
- Brunland,Lingjærde & Maus: Rett på Java
- Lemay/Cadenhead: SAMS Teach Yourself Java 2 platform in 21 days.

Sensurdato:
7. sept. 2005. Resultater gjøres kjent på http://studweb.ntnu.no/ og sensurtelefon 81 54 80 14.

Prosentsatser viser hvor mye hver oppgave teller innen settet.

Merk: All programmering skal foregå i Java.

Lykke til!

OPPGAVE 1 (10%): Iterasjon i tabeller og List.

a) Skriv kode for en metode `Object[] reverser1(Object[] tabell)` som returnerer en kopi av `tabell` med alle elementene i *motsatt* rekkefølge. Mao. skal første element i `tabell` være sist i resultattabellen, element nr. 2 i `tabell` skal være nest sist, osv.

b) Skriv kode for en metode `void reverser2(List liste)` som reverserer `liste`, slik at rekkefølgen snus. Mao. det elementet i `liste` som var først før et kall til `reverser2` skal være sist etter kallet, det elementet i `liste` som var nr. 2 skal være neste sist osv. To kall etter hverandre vil gjenopprette den originale rekkefølgen. Relevante metoder i `List`-grensesnittet (som `ArrayList` implementerer):

```
// Appends the specified element to the end of this list.
boolean add(Object element)

// Inserts the specified element at the specified position.
void add(int index, Object element)

// Removes the element at the specified position in this list.
// Returns the removed element.
Object remove(int index)

// Removes the first occurrence in this list
// of the specified element.
Boolean remove(Object element)

// Returns the element at the specified position in this list.
Object get(int index)

// Replaces the element at the specified position in this list
// with the specified element.
Object set(int index, Object element)

// Returns the number of elements in this list.
int size()
```

OPPGAVE 2 (15%): Klasse og Iterator.

a) Skriv kode for en Teller-klasse, som har et teller-attributt av typen int og en tell-metode. Tell-metoden skal øke teller-attributtet med 1 hver gang den kalles.

b) Utvid Teller-klassen med relevante attributter og konstruktør(er), slik at teller-attributtet *starter på* en bestemt start-verdi og *stopper på* en bestemt slutt-verdi (dvs. når slutt-verdien, men telles deretter ikke videre).

c) Skriv kode for nødvendig metoder, slik at Teller-klassen implementerer Iterator og returnerer tallene *fra og med* startverdien og *opp til, men ikke inkludert* slutt-verdien, én etter én for hvert kall til next-metoden. Kall eksisterende metoder, der du kan. Metodene i Iterator-grensesnittet er som følger:

```
// Returns true if the iteration has more elements.
boolean hasNext()

// Returns the next element in the iteration.
Object next()
```

d) Utvid implementasjonen av next-metoden slik at den kaster et unntak av typen NoSuchElementException, dersom den kalles etter at slutt-verdien er nådd.

OPPGAVE 3 (45%): Klasser og Collection-klasser.

I denne oppgaven skal du implementere en enkel kontaktbok med telefonnumre i.

a) Forklar prinsippet bak innkapsling og bruken av nøkkelordene `private` og `public`.

b) Skriv kode for en klasse Nummerliste, som skal brukes for å holde et sett telefonnumre, av typen String. Klassen skal ha et attributt kalt `numre` av typen ArrayList deklarert som `private` og en konstruktør som initialiserer `numre`-attributtet. Implementer følgende innkapslingsmetoder:
   - getAntallNumre skal returnere antall numre
   - getNummer skal returnere et gitt nummer
   - setNummer skal endre et gitt nummer
   - addNummer skal legge til et nytt nummer

c) I en alternativ implementasjon av Nummerliste kunne vi *arvet fra* ArrayList, istedenfor å ha et ArrayList-attributt. Forklar hvordan metodene du hittil har implementert i tilfelle måtte endres. Hva er ulempen med å arve fra ArrayList på denne måten?

d) Det er vanlig i kontaktbøker at numre kategoriseres, som f.eks. ”hjemme”, ”mobil”, ”jobb” osv. Du skal skrive kode for en subklasse av Nummerliste kalt KategorisertNummerliste, som deklarerer et nytt attributt ved navn `kategorier` av typen ArrayList. Elementene i katetorier skal tilsvare dem i numre, slik at hvert nummer i numre-lista har sin kategori i samme posisjon i kategorier-lista.

   1) Skriv kode for setNummer- og addNummer-metoder tilsvarende dem i deloppgave b), som tar inn et ekstra kategoriparameter.
   2) Redefiner den eksisterende addNummer-metoden, altså den uten katagoriparameter, slik at den setter kategorien til null.
   3) Skriv kode for en finnKategori-metode, som tar inn et nummer (en String) og returnerer tilsvarende kategori i kategorier-lista eller null hvis nummeret ikke finnes numre-lista.

OPPGAVE 4 (15%): Testing

a) Beskriv en generell testemetodikk, såkalt *enhetstesting*, som lar en teste *oppførselen til* Teller-klassen fra oppgave 2, uten at en nødvendigvis har skrevet koden selv eller har den tilgjengelig. Formuler 2 relevante regler for oppførselen til Teller-klassen.

b) Skriv kode for en eller flere JUnit-testmetoder som tester alle metodene i Teller-klassen fra oppgave 2, både metodene skrevet frem til og med deloppgave 2b) og de to skrevet i deloppgave 2c). Bruk metodene `assertEquals(Object, Object)` og `assertTrue(boolean)` i TestCase for å sjekke relevante verdier.

c) Forklar hvordan en kan teste at en metode, f.eks. Teller sin next-metode fra deloppgave 2d), kaster et unntak i riktig tilfelle og av riktig type.

OPPGAVE 5 (15%): Observatør-observert-teknikken

Du ønsker å gjøre Nummerliste-klassen fra oppgave 3 *observerbar*, i den forstand at den kan si fra til en eller flere *observatører* at et eksisterende nummer er endret eller at et nytt er lagt til. Gitt følgende lyttergrensesnitt:

```
public interface Nummerlistelytter {
   // Kall til  listeEndret-metoden angir at endretListe er endret
   // i posisjon indeks.
   public void listeEndret(Nummerliste endretListe, int indeks);
}
```

a) Anta at en Nummerliste kan ha én eller flere Nummerlistelyttere knyttet til seg. Hvordan må du endre Nummerliste for at den skal fungere som observert og si fra om endringene?

b) Tegn sekvensdiagram som viser hvordan en observatør (et tenkt objekt) legger seg på som lytter, kaller en metode på et Nummerliste-objekt og får melding om at listen er endret.

c) Forklar hvordan du kan bruke en JUnit-testklasse til å teste at Nummerliste implementerer rollen som observert på riktig måte, dvs. kaller listeEndret-metoden på observatørene sine på riktig tidspunkt og med riktige parametre.