NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

EKSAMEN I FAG
TDT4100 Objektorientert programmering

Fredag 2. juni 2006
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:

- Én og kun én trykt bok, f.eks. Lewis & Loftus: Java Software Solutions

Sensurdato:
23. juni 2006. 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 (15%): Iterasjon.

a) Skriv kode for en metode `Object[] flett1(Object[] tabell1, Object[] tabell2)` som returnerer en tabell med alle elementene fra `tabell1` og `tabell2` *flettet*. Anta at `tabell1` er `[x1, x2, ..., xN]` og `tabell2` er `[y1, y2, ..., yN]`. Da skal resultat-tabellen være `[x1, y1, x2, y2, ..., xN, yN]`. Du kan anta at `tabell1` og `tabell2` er like lange.

b) Skriv kode for en metode `List flett2(Iterator it1, Iterator it2)` som returnerer en ny `List` med elementene fra `it1` og `it2` flettet. Annenhvert element i resultat-lista skal komme fra `it1` og annenhvert fra `it2`, *inntil en av dem er tømt*. Anta at `it1` genererer `[x1, x2, ..., xN]` og `it2` genererer `[y1, y2, ..., yN+2]`. Da skal resultat-lista inneholde `[x1, y1, x2, y2, ..., xN, yN]`

c) Skriv kode for en metode `List flett3(Iterator it1, Iterator it2)` som returnerer en ny `List` med elementene fra `it1` og `it2` flettet, som i b), men hvor alle elementene fra `it1` og `it2` brukes. Anta at `it1` genererer `[x1, x2, ..., xN]` og `it2` genererer `[y1, y2, ..., yN+2]`. Da skal resultat-lista inneholde `[x1, y1, x2, y2, ..., xN, yN, yN+1, yN+2]` Du trenger ikke gjenta uendret kode fra `flett2`, så lenge det går klart frem hva som er uendret og hva som endres/erstattes.

Relevante metoder fra `List`-grensesnittet:

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

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

// Returns the index in this list of the first occurrence
// of the specified element, or -1 if it doesn't exist in this list.
int indexOf(Object element)
```

Relevante metoder fra Iterator-grensesnittet:

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

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

OPPGAVE 2 (40%): Enkle klasser

Du skal lage klasser for å beskrive en CD og musikken som er på den. Klassene skal brukes i en applikasjon for CD-brenning, hvor en skal kunne fylle en eller flere CD'er med musikk,

før de(n) brennes. Merk at ikke alle deloppgaver krever at de foregående er løst, så ikke hopp over de resterende deloppgavene om én blir for vanskelig. Bruk gjerne grensesnitt og klasser fra Java sitt API/klassebibliotek, f.eks. Collection-rammeverket.

a) Hver CD har et navn og inneholder et sett spor, og hvert spor har et navn, en ventepause angitt i hele sekunder og en spillelengde angitt i tidels sekunder. Definer klassene CD og Spor og feltene som trengs for å holde disse dataene.

b) Forklar begrepet *innkapsling*.

c) Skriv kode for nødvendige metoder i CD og Spor for å kapsle inn feltene definert i a). Forklar hvordan du velger navn, retur- og parametertyper for disse metodene.

d) Lag en metode computePlayTime i CD-klassen som tar inn en logisk verdi og som returnerer tiden i antall sekunder (med desimaler) det tar å spille av CD'en. Den logiske verdien skal avgjøre om ventepausene skal regnes med (logisk verdi er sann) eller ikke (logisk verdi er usann).

e) Du ønsker å lage en hjelpemetode `sorter` i Spor-klassen som sorterer en tabell med Spor-objekter, basert på musikklengden. Skriv metodedeklarasjonen og forklar hvordan du vil implementere `sorter`-metoden vha. klasser/grensesnitt og metoder definert i Collection-rammeverket. Merk at du trenger ikke implementere noen sorteringsalgoritme selv.

En CD kan ha plass til maksimum 72 minutter med musikk (altså unntatt ventepausene mellom sangene) og det er viktig å unngå at brukeren prøver å brenne CD'er med for liten plass til innholdet.

f) Definer en type unntak for å si fra om at en CD er i ferd med å bli overfylt. Utvid relevante metoder fra b) slik at det genereres/kastes et unntak av denne typen dersom en prøver å legge Spor til en CD slik at musikklengden overstiger 72 minutter.

g) Hva skjer med din implementasjon frem til og med f), dersom musikklengden til et Spor-objekt økes *etter* at den er lagt til CD-objektet? Forklar (kode er ikke nødvendig) hvordan du vil endre CD og/eller Spor for å sikre at det også da kastes unntak om den nye musikklengden er for stor.

h) Anta at du skal støtte både vanlige CD'er og mini-CD'er, hvor forskjellen er at de rommer ulik mengde musikk (målt i spilletid og ikke antall spor). Du innfører derfor en ny klasse MiniCD, og ønsker å ha *mest mulig kode* samlet i en felles abstrakt klasse med navn AbstractCD, som CD- og MiniCD-klassene arver fra. Forklar hvordan MiniCD- og AbstractCD-klassene blir og hvordan du vil strukturere om og evt. endre koden du allerede har skrevet.

OPPGAVE 3 (30%): Collection-rammeverket.

I denne oppgaven skal du jobbe med en implementasjon av en forenklet java.util.Map (heretter bare kalt Map). Det er *ikke* lov til å bruke klasser fra Collection-rammeverket i denne oppgaven. Det kan være lurt å lese gjennom hele oppgaven før du bestemmer deg for hvordan du løser hver enkelt del.

Map-grensesnittet er definert som følger:

```
public interface Map {
   // size returnerer antall nøkkel/verdi-par.
   public int size();

   // put knytter value til nøkkelen key,
   // slik at get(key) returnerer value.
   // En evt. eksisterende verdi for key blir erstattet.
   // Dersom verdien er null, skal nøkkel/verdi-paret fjernes.
   public void put(Object key, Object value);

   // get returnerer verdien knyttet til key,
   // eller null dersom nøkkelen ikke finnes.
   public Object get(Object key);

   // keys returnerer en tabell med alle nøklene.
   public Object[] keys();
}
```

Anta at følgende hjelpeklasse allerede er implementert:

```
public class ArrayUtil {
   // Returns a new array containing all the objects from os and o
   public static Object[] add(Object[] os, Object o)

   // Returns a new array containing
   // all the objects from os except the one at position indeks
   public static Object[] remove(Object[] os, int indeks)
}
```

a) Skriv kode for klassen ArrayMap som implementerer Map-grensesnittet og som bruker én eller flere tabeller for å holde nøklene og verdiene.

b) Skriv kode for en konstruktør ArrayMap(Map annenMap) som initialiserer det nye ArrayMap-objektet med nøkkel/verdi-parene fra annenMap.

c) Når en sammenligner nøkler i ArrayMap sine metoder, har en valget mellom (minst) to måter å sammenligne objekter (for likhet) på. Hvilke to er det, hva er forskjellen og evt. fordelen/ulempen med hver av disse? Hvordan kan du skrive (om) ArrayMap slik at det er lett å bytte (til en annen) sammenligningsmetode?

OPPGAVE 4 (15%): Testing

a) Definer *regler for oppførselen* til metodene i Map-grensesnittet definert i oppgave 3. Forklar hvordan en på grunnlag av slike regler kan skrive testkode som tester at en spesifikk Map-implementasjon (f.eks. ArrayMap) følger disse reglene. Merk at det er den generelle teknikken vi ønsker forklaring på, ikke spesifikt JUnit-rammeverket.

b) Skriv en eller flere testmetoder i en tenkt TestCase-subklasse med JUnit-rammeverket, som til sammen tester get-, put- og size-metodene i ArrayMap-implementasjonen av Map-grensesnittet. Bruk metodene assertEquals(Object, Object), assertTrue(boolean) og assertNull(Object) i TestCase for å sjekke relevante verdier.