Tentamen Programmeringsteknik I 2012-10-25

Skrivtid: 1400-1700
Hjälpmedel: Java-bok

Tänk på följande

- Det finns en referensbok (Java) hos tentavakten som du får gå fram och läsa men inte ta tillbaka till bänken.
- Skriv läsligt! Använd *inte* rödpenna!
- Skriv bara på framsidan av varje papper.
- Lägg uppgifterna i ordning. Skriv uppgiftsnummer och pin-kod (eller namn om du saknar sådan) på alla papper. Skriv inte längst upp i vänstra hörnet - det går inte att läsa där efter sammanhäftning.
- Fyll i försättssidan fullständigt.
- Det är principer och idéer som är viktiga. Skriv så att du övertygar examinator om att du har förstått dessa även om detaljer kan vara felaktiga.
- Programkod skall vara läslig dvs den skall vara vettigt strukturerad och indenterad. Namn på variabler, metoder, klasser etc skall vara beskrivande men kan ändå hållas ganska korta.
- Det är totalt 40 poäng på skrivningen. Betygsgränser: 18 ger säkert 3, 26 ger säkert 4, 34 ger säkert 5.

Lycka till!

Anna och Tom

Inledning

I ett bibliotek brukar böckerna (bland annat) sorteras efter författare och titel. Att hålla böckerna sorterade innebär dock en hel del arbete. Ett alternativ vore att bara placera in böckerna i första bästa hylla som har plats och låta ett datorsystem hålla reda på var böckerna finns. När man vill låna en bok ger man dess författare och titel och erhåller då vilken hylla och var på hyllan boken finns. När man återlämnar boken ställer man boken sist på den hylla systemet anger.

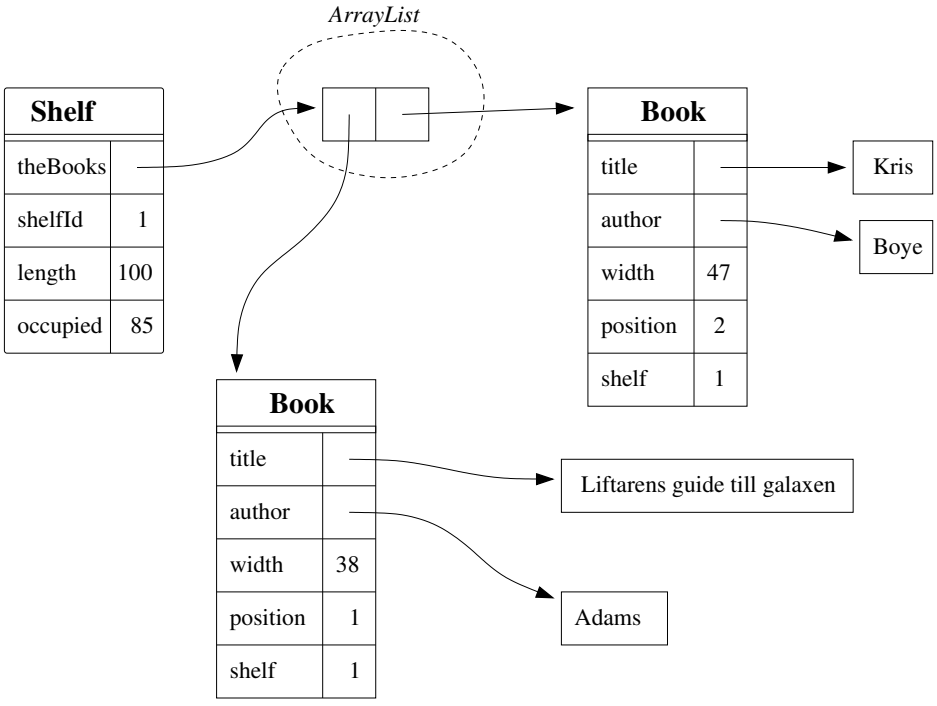
I bilagan finns ett ofullständigt Java-program som hjälper till med detta.

Programmet har tre klasser: `Book`, `Shelf` och `Library`.

Klassen `Book` representerar ett exemplar av en given bok (det kan finnas flera exemplar av samma bok). Varje bok-objekt skall ha en *författare*, en *titel* och en *bredd* som anger hur många mm boken tar av ett hyllplan. Slutligen finns det information vilken hylla (hyllans id-nummer) och exemplarets ordningsnummer ("position") på hyllan (boken längst till vänster har position 1, nästa position 2 etc.) Position 0 anger att boken inte har någon plats tilldelad.

Klassen `Shelf` representerar ett hyllplan som innehåller att antal `Book`-objekt. Antalet böcker som ryms på ett hyllplan beror dels på hyllplanets längd och dels på böckernas bredd. Eftersom antalet är obekant och varierar är det lämpligt att representera innehållet med en `ArrayList`. Hyllplanet håller reda på hur mycket av utrymmet (i mm) på hyllan som är upptaget. Varje hyllplan har vidare ett *id-nummer*.

Ett hyllplan med två bokobjekt kan illustreras med följande figur:

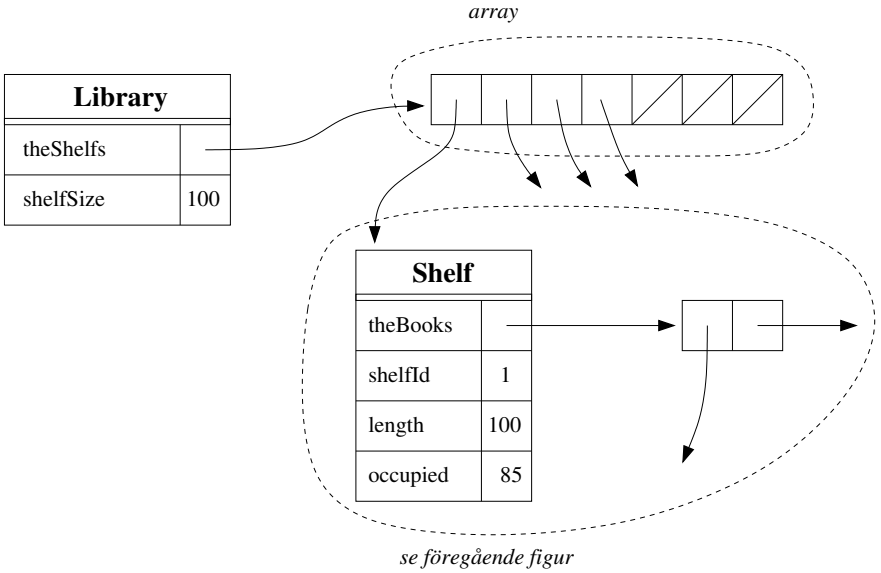


Anmärkning: Måtten (hyllängd och bokbredd) är inte så verklighetstrogna - de är valda för att körexemplet längre ned skall kunna illustrera vad som händer på ett enkelt sätt.

Klassen `Library` representerar ett bibliotek. Ett bibliotek består av ett antal hyllor. Ett

bibliotek byggs för ett visst maxantal hyllor. Själva hyllorna skaffas successivt allteftersom behov uppstår (dvs när man inte hittar plats för en bok). Eftersom det finns ett maxantal hyllor och vi inte behöver någon dynamik använder vi en *array* för att representera samlingen av hyllor. Vi tänker oss (åtminstone tills vidare) att biblioteket alltid skaffar hyllor av en standardlängd och låter således denna längd vara ett attribut i klassen `Library`.

Ett bibliotek med plats för 7 hyllor och med 4 anskaffade hyllar kan illustreras med följande figur:



Uppgifter

1. I klassen `Book`:

   a) Skriv klar konstruktorn `public Book(String author, String title, int width)` som skapar ett bok-objekt utan att tilldela det någon platsinformation. (2p)

   Svar:
   ```
   public Book(String author, String title, double width) {
     this.title = title;
     this.author = author;
     this.width = width;    // in mm
     position = 0;            // för tydlighetens skull - ej nödvändig
     shelf = 0;               // för tydlighetens skull - ej nödvändig
   }
   ```

   b) Skriv klar `toString`-metoden som returnerar författare och titel. Om det finns platsinformation (dvs om `position` är större än 0) skall platsinformationen (hylla och position — se körexemplen) inkluderas. (2p)

   Svar:
   ```
   public String toString() {
     String r = author + ": " + title;
     if ( position>0 ) {
       r = r + "   <" + shelf + ", " + position + ">";
     }
     return r;
   }
   ```

   c) Skriv klar metoden `boolean equals(Book b)` som returnerar `true` om denna bok har samma titel och författare som boken `b`, annars `false`. (2p)

Svar:
```
public boolean equals(Book b) {
  return author.equals(b.author) && title.equals(b.title);
}
```

2. I klassen `Shelf`:

   a) Skriv klar metoden `int getFreeSpace()` som returnerar hur mycket ledigt utrymme som finns på hyllan. (2p)

   Svar:
   ```
   public int getFreeSpace() {
     return length - occupied;
   }
   ```

   b) Skriv klar metoden `boolean add(Book b)` som, om det går, lägger in boken `b` som sista bok på hyllan. Metoden måste underhålla såväl hyllans som bokens instansvariabler. Om det gick att ställa boken på hyllan skall metoden returnera `true` annars `false`. (4p)

   Svar:
   ```
   public boolean add(Book b) {
     if (b.getWidth() > getFreeSpace()) {
       return false;
     } else {
       b.setPosition(theBooks.size()+1);
       b.setShelf(shelfId);
       theBooks.add(b);
       occupied += b.getWidth();
       return true;
     }
   }
   ```

   c) Skriv klar metoden `ArrayList<Book> findByAuthor(String author)` som skapar och returnerar en lista över de exemplar av böcker skrivna av angiven författare som finns på denna hylla. (4p)

   Svar:
   ```
   public ArrayList<Book> findByAuthor(String author)  {
     ArrayList<Book> res = new ArrayList<Book>();
     for (int i= 0; i<theBooks.size(); i++) {
       Book b = theBooks.get(i);
       if (author.equals(b.getAuthor())) {
         res.add(b);
       }
     }
     return res;
   }
   ```

   d) Skriv klar metoden `Book remove(int position)`. Metoden skall kontrollera att `position` har ett legalt värde och därefter ta bort boken på denna position. Metoden måste uppdatera positionsangivelserna för kvarvarande böcker liksom även informationen om upptaget utrymme. (4p)

Svar:
```
public Book remove(int position) {
   if (position<1 || position>theBooks.size()) {
     System.out.println("Shelf " + shelfId +
                        " has no book at position " + position);
     return null;
   }
   Book ret = theBooks.remove(position-1);
   ret.setPosition(0);
   occupied = occupied - (int)ret.getWidth();
   // Update the positions for the books after the removed book
   for (int i=position-1; i<theBooks.size(); i++) {
     Book b = theBooks.get(i);
     b.setPosition(i+1);
   }
   return ret;
 }
```

3. I klassen `Library`:

   a) Skriv klar konstruktorn `Library(int librarySize, int shelfSize)` som skapar ett bibliotek med plats för `librarySize` hyllor, alla av storlek `shelfSize`. (2p)

   Svar:
   ```
   public Library(int librarySize, int shelfSize) {
     theShelfs = new Shelf[librarySize];
     this.shelfSize = shelfSize;
     onLoan = new ArrayList<Book>();   //  För användning i uppgift 3 c
   }
   ```

   b) Skriv klar metoden `boolean add(Book book)` som letar upp en hylla med plats för boken `book` och lägger in den där. Om boken inte får plats på någon existerande hylla skall en ny skapas. Om det inte går att skapa fler hyllor skall ett felmeddelande ges och `false` returneras. Om metoden lyckades lagra boken skall `true` returneras. (5p)

   Svar:
   ```
   public boolean add(Book book) {
     // Find a shelf with enough space
     for (int i=0; i<theShelfs.length; i++) {
       if ( theShelfs[i]==null ) {
         theShelfs[i] = new Shelf(shelfSize, i+1);
       }
       if (theShelfs[i].getFreeSpace()>book.getWidth()) {
         theShelfs[i].add(book);
         return true;
       }
     }
     System.out.println("Library full! Build a new one!");
     return false;
   }
   ```

   c) Koden i det givna systemet håller inte reda på utlånade böcker. Detta kan dock åtgärdas genom några kompletteringar i klassen `Library`. Vilka? Skriv den kod som behövs! (7p)

   Svar: Lägg till en instansvariabel `ArrayList<Book> onLoan` som innehåller utlånade exemplar. Skriv metoderna `borrowBook(int shelf, int position)` och `returnBook(String author, String title)`.
   (Parametrarna till `borrowBook` skulle också kunna vara `author` och `title`.)

Kod:

```
public Book borrowBook(int shelf, int position) {  // 3 c
   Book b = remove(shelf, position);
   System.out.println("Borrowed: " + b);
   onLoan.add(b);
   return b;
}

public void returnBook(String author, String title) {   // 3 c
  Book b = new Book(author, title, 0);
  for(int i = 0; i<onLoan.size(); i++) {
    if (b.equals(onLoan.get(i))) {
      add(onLoan.get(i));
      onLoan.remove(i);
      return;
    }
  }
  System.out.println("Not borrowed from this library: " + b);
  }
```

4. Besvara *kortfattat* följande uppgifter/frågor

   a) Vad är en *instansvariabel*?

   b) Vad är en *metod*?

   c) Vad menas med *iteration*?

   d) Vad är en *konstruktor*?

   e) Vad sker när ett Java-program *kompileras*?

   f) Vad är det för skillnad på *formella* och *aktuella* parametrar?

(6p)