DAT043 – Objektorienterad programmering för D, DIT011 – Objektorienterad programvaruutveckling för GU

tentamen 2017-08-16

Tid: 8:30 - 12:30. Plats: SB.
Ansvarig lärare: Fredrik Lindblad, tel.nr. 031-772 2038. Besöker tentamenssalarna cirka 9:30 och cirka 11:00.
Maxpoäng: 60. Betygsgränser Chalmers: 24 för 3:a, 36 för 4:a, 48 för 5:a. Betygsgränser GU: 24 för G, 48 för VG.
Tentamen består av 6 uppgifter.
Tillåtna hjälpmedel: Den dubbelsidiga lathund som finns tillgänglig på kurssidan och upptryckt vid tentamen. Inga egna anteckningar på detta blad.

Implementeringar ska skrivas i Java. Oväsentliga syntax- och namnfel ger inte poängavdrag.
Skriv tydligt och välstrukturerat. Koden behöver inte kommenteras. Svårförståeliga lösningar kan underkännas helt eller ges poängavdrag.
Lösningar som är onödigt krångliga eller inte följer god programmeringsstil, dels allmänt och dels med tanke på idéerna med objektorienterad programmering, kan underkännas helt eller ges poängavdrag.
Om det inte uttryckligen står motsatsen i uppgiften kan du använda klasser och metoder i Java:s API.
Om det inte uttryckligen står motsatsen i uppgiften får du definiera egna hjälpmetoder.
Importeringar av klasser i Java:s API behöver inte skrivas ut.
Svara inte på flera uppgifter på samma blad. Om en uppgift har deluppgifter så svara gärna på flera deluppgifter på samma blad.
Efter rättning är tentamen tillgänglig på expeditionen på plan 4 i EDIT-huset och kan granskas där. Den som vill diskutera rättningen kan kontakta ansvarig lärare och boka tid för ett möte. Notera att tentamen i så fall inte ska tas med från expeditionen.

---

1. (a) Givet klassen `C` (se nedan), vad skrivs ut om man exekverar metoden `m`?

```
public class C {
  static int x;
  public C(int y) { x = y; }
  public int v() { return x; }
}

public static void d(String s, String[] a) {
  for (int i = 0; i < a.length; i++) {
    a[i] = s;
  }
}
public static void m() {
  String m = "M";
  String[] arr = new String[2];
  d(m, arr);
  m = m + "Z";
  C c1 = new C(6);
  C c2 = new C(8);
  System.out.println(arr[0] + arr[1] + c1.v() + c2.v());
}
```

(4p)

(b) Givet är följande klasser och metoder:

```
class A { ... }
class B extends A { ... }
class C extends B { ... }
public static B f(A x) { ... }
public static C g(B x) { ... }
```

Definitionerna är irrelevanta och därför utelämnade, men klasserna har alla en publik konstruerare som inte tar några argument.

Vilka rader i följande kodutsnitt innehåller fel?

```
C x1 = new A();
B x2 = new C();
C x3 = new B();
A x4 = new C();
A x5 = f(new B());
C x6 = g(new A());
A x7 = new B();
C x8 = (C)x7;
B x9 = (B)x7;
```

Ange vilken eller vilka rader som innehåller fel och för varje felaktig rad om felet uppstår vid kompilering eller exekvering av programmet. Använd numret på variabeln som deklareras när du anger rader, d.v.s. ett tal mellan 1 och 9. Du ska utgå från att `A`, `B`, `C`, `f` och `g` alla är tillgängliga. Uppgiften handlar alltså inte om huruvida dessa är `private`, `public`, `static` etc.

(4p)

2. Skriv en klass som implementerar ett program `Palindrom` som läser in en textfil innehållande ord separerade med mellanslag. När man startar programmet skall man på kommandoraden ange ett argument; filnamnet för textfilen. Man kan t.ex. ge kommandot

```
java Palindrom ord.txt
```

Programmet ska skriva ut alla ord i filen som är palindrom, d.v.s. utgörs av samma följd av bokstäver både framifrån och bakifrån. Programmet ska inte skilja på stora och små bokstäver, d.v.s. `'A'` och `'a'` ska ses som samma bokstav. Programmet behöver inte hantera skiljetecken (punkt, komma, etc.). Om t.ex. filen `ord.txt` har innehållet

```
Otto sa tut i sås igår
```

så ska kommandot ovan ge utskriften

```
Otto tut i sås
```

Programmet skall kontrollera att antalet argument är korrekt och att filerna går att öppna. Om något skulle vara fel skall en felutskrift ges och programmet avslutas.

(10p)

3. Implementera den statiska, generiska metoden

```
public static <E> int antalUnika(E[] a)
```

Metoden ska returnera antalet unika värden i arrayen `a`. Huruvida två värden är lika definieras av instansmetoden `equals` som hör till den konkreta typ för vilken metoden anropas.

Om t.ex. `a` är

```
{2, 5, 3, 7, 4, 3, 5, 4, 2, 1}
```

så ska metoden returnera `6` eftersom det finns sex unika värden i arrayen; 1, 2, 3, 4, 5 och 7.

Metoden ska inte förändra elementen i `a`.

(8p)

4. Du ska skriva tre klasser som implementerar ett förenklat system för böcker i ett bibliotek. Klassen `Bok` representerar en utgiven bok. En instans av `Bok` representerar inte en faktiskt kopia av boken. Istället representeras en kopia av en bok av klassen `Exemplar`. Varje `Bok` är associerad till en uppsättning `Exemplar` och varje `Exemplar` är associerad med en unik `Bok`. Den tredje klassen, `LånbartExemplar`, representerar en kopia som biblioteket tillåter utlåning av. Denna klass ska ärva `Exemplar`.

   Klassen `Bok` ska ha konstrueraren och metoderna

   `public Bok(String titel)` som skapar en ny bok med en angiven titel och utan associerade exemplar.

   `public String titel()` som returnerar bokens titel.

   `public void läggTillExemplar(Exemplar ex)` som associerar en exmplar till boken. Om den bok som exemplaret är associerat med inte är aktuell instans så ska ett undantag av typen `IllegalArgumentException` kastas.

   `public Exemplar tillgängligtExemplar()` vilken returnerar ett exemplar associerat till boken som är tillgängligt för utlåning, d.v.s. kan lånas ut men är ej utlånat. Om inget sådant exemplar finns så returneras `null`.

   Klassen `Exemplar` ska ha konstrueraren och metoden

   `public Exemplar(Bok bok)` som skapar ett nytt exemplar av angiven bok.

   `public Bok bok()` som returnerar boken som instansen representerar ett exemplar av.

   Klassen `LånbartExemplar` ska ha konstrueraren och metoderna

   `public LånbartExemplar(Bok bok)` som skapar ett nytt lånbart exemplar av angiven bok. Exemplaret ska initieras till ej utlånat tillstånd.

   `public void lånaUt()` som försätter exemplaret i utlånat tillstånd.

   `public void återfå()` som försätter exemplaret i ej utlånat tillstånd.

   `public boolean utlånad()` som returnerar `true` om exemplaret är utlånat och `false` annars.

   De instansvariabler du väljer att förse klasserna med ska alla vara privata.

(12p)

5. Skriv en java-klass `Poly` som representerar ett muterbart polynom ($p(x) = m_0 + m_1 \cdot x + m_2 \cdot x^2 + \cdots + m_n \cdot x^n$). Den ska ha en konstruerare `public Poly()` som skapar en instans som representerar det konstanta noll-polynomet ($p(x) = 0$).

   Vidare ska det finnas en metod

   ```
   public void addTerm(double m, int k)
   ```

   som adderar termen $m \cdot x^k$ till polynomet. För att skapa en instans som motsvarar $p(x) = 5 \cdot x^2 + 3 \cdot x + 7$ så kan man skriva följande kod:

   ```
   Poly p = new Poly(); p.addTerm(5, 2); p.addTerm(3, 1); p.addTerm(7, 0);
   ```

   Ordningen som termerna adderas ska inte spela någon roll och man ska kunna lägga till termer med samma exponent flera gånger. Följande kod ska alltså ge en instans för samma polynom som ovan:

   ```
   Poly p = new Poly(); p.addTerm(7, 0); p.addTerm(1, 2); p.addTerm(3, 1); p.addTerm(4, 2);
   ```

   Klassen ska också ha en metod

   ```
   public double eval(double x)
   ```

   som beräknar polynomets värde i punkten `x`, d.v.s. $p($`x`$)$ där $p(x)$ är det polynom som instansen representerar.

   Slutligen ska klassen implementera likhetstest:

   ```
   public boolean equals(Object obj)
   ```

   Denna metod ska returnera `true` om `obj` är en instans av `Poly` som motsvarar samma polynom som aktuellt objekt.

   (12p)

6. Implementera metoden

   ```
   public static void copy(int[] a, int srcStart, int srcEnd, int dstStart)
   ```

   som kopierar ett intervall av element i `a` till ett annat ställe i arrayen. Mer specifikt ska `a[srcStart` $+ i$`]` kopieras till `a[dstStart` $+ i$`]` för alla i från och med 0 till och med `srcEnd` $-$ `srcStart` $- 1$.

   Om t.ex. `a` är

   ```
   {11, 12, 13, 14, 15, 16, 17, 18}
   ```

   så har arrayen efter anropet `copy(a, 2, 4, 5)` ändrats till

   ```
   {11, 12, 13, 14, 15, 13, 14, 18}
   ```

   Metoden ska hantera situationen att käll-intervallet överlappar mål-intervallet. Om `a` åter är `{11, 12, 13, 14, 15, 16, 17, 18}` så ska arrayen efter anropet `copy(a, 1, 5, 3)` ändrats till

   ```
   {11, 12, 13, 12, 13, 14, 15, 18}
   ```

   Metoden behöver inte hantera fallen då `srcEnd` $<$ `srcStart`, `srcStart` < 0, `srcEnd` $>$ `a.length`, `dstStart` $<$ 0 eller `dstStart` $+$ `srcEnd` $-$ `srcStart` $>$ `a.length`.

   (10p)