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

tentamen 2017-06-09

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) Vad skrivs ut om man exekverar metoden `g` i programkoden nedan?

```
public static void f(int[] a, int[] b) {
  if (a.length != b.length) throw new IllegalArgumentException();
  int[] t = new int[a.length];
  for (int i = 0; i < a.length - 1; i++) {
    t[i] = a[i];
    a[i] = b[i];
  }
  b = t;
}
public static void g() {
  int[] x = {2,5,8};
  int[] y = {6,3,7};
  f(x, y);
  System.out.println(Arrays.toString(x) + " " + Arrays.toString(x));
}
```

Tips: Om `z` är arrayen `{1,2,3,4}` så returnerar `Arrays.toString(z)` strängen `"[1, 2, 3, 4]"`.

(4p)

(b) Givet är följande gränssnitt, klasser och metoder:

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

Definitionerna är irrelevanta och därför utelämnade.

Vilka rader i följande kodutsnitt innehåller fel?

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

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 8. 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 `Count` som läser in en textfil innehållande heltal separerade med mellanslag. När man startar programmet skall man på kommandoraden ange ett argument; filnamnet för textfilen. Man kan t.ex. ge kommandot

```
java Count numbers.txt
```

Programmet ska räkna ut hur många gånger varje heltal förekommer. Programmet ska, till standard output, skriva ut varje heltal som förekommer i filen tillsammans med antalet förekomster. Om filen `numbers.txt` t.ex. har innehållet

```
43 85 62 36 62 12 85 324 85
```

så ska kommandot ovan ge utskriften

```
36 -> 1
324 -> 1
85 -> 3
43 -> 1
12 -> 1
62 -> 2
```

Ordningen som de förekommande heltal skrivs ut i spelar ingen roll. Det viktiga är att alla tal som förekommer (och inga andra) skrivs ut och att antalet som anges är korrekt.

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 extends Comparable<E>> E minimum(E[] arr)
```

Det framgår av signaturen att metoden kan tillämpas på arrayer för vars element en ordning finns definierad. Metoden ska returnera det minsta elementet i arrayen. Om `arr` är `null` eller en tom array ska metoden returnera `null`.

Du ska inte använda metoder och klasser i Java:s standard-API, förutom gränssnittet `Comparable`.

(8p)

4. Skriv tre java-klasser som implementerar följande gränssnitt:

```
interface GrObj {
  abstract public void draw(Graphics g, int x, int y);
}
```

Detta är tänkt att vara ett gränssnitt för grafisk objekt som kan ritas i en 2D-kontext (ett `Graphics`-objekt). Objekt av typen `GrObj` har en form, men ingen position. Positionen, eller närmare bestämt objektets övre vänstra hörn på ritytan, anges istället när man anropar `draw`. Denna metod ska rita rätt form på angiven position.

Den första klass som ska implementera `GrObj` är `Rectangle`. Den ska motsvara en rektangel med en bredd och en höjd (mätt i antal pixlar). Den ska ha en konstruktor

```
public Rectangle(int w, int h)
```

som skapar ett rektangel-objekt med bredden `w` och höjden `h`. Koden som ritar ut en rektangel kan använda instansmetoden `drawRect` i `Graphics`. Argumenten `x` och `y` (se lathund) i denna metod motsvarar övre vänstra hörnet.

Den andra klassen är `Circle` som motsvara en cirkel. Den ska ha en konstruktor

```
public Circle(int diam)
```

som skapar ett cirkel-objekt med diametern `diam`. Koden som ritar ut en cirkel kan använda instansmetoden `drawOval` i `Graphics`. Argumenten `x` och `y` (se lathund) i denna metod motsvarar övre vänstra hörnet.

Den tredje och sista klassen är `Group` och denna ska motsvara en grupp av godtyckligt antal objekt med tillhörande position. Den ska ha en konstruktor

```
public Group()
```

som skapar ett tomt grupp-objekt. Den ska också ha en metod

```
public void add(GrObj o, int x, int y)
```

som lägger till ett `GrObj`-objekt `o` med positionen angiven av `x` och `y`. När ett objekt av typen `Group` ritas ut ska alla objekten som tillhör gruppen ritas ut, relativt den, vid anropet till `draw`, angivna positionen.

Ett exempel:

```
Rectangle rect = new Rectangle(20, 10);
Circle circ = new Circle(30);
Group group = new Group();
group.add(rect, 50, 100);
group.add(circ, 120, 80);
group.draw(g, 35, 65);
```

Givet ett giltigt objekt `g` av typen `Graphics` så skulle denna kod rita ut en rektangel med bredd 20 och höjd 10 på position $(85, 165)$ och en cirkel med diameter 30 på position $(155, 145)$.

(12p)

5. Skriv en java-klass `Interval` som representerar slutna intervall på den reella (approximerat med flyttal) talaxeln. Ett intervall $[x, y]$ där $x \leq y$ motsvarar mängden av alla tal z sådana att $x \leq z \leq y$. Ett intervall $[x, y]$ där $x > y$ motsvarar den (unika) tomma mängden.

   Klassen ska ha en konstruktor

   ```
   public Interval(double x1, double x2)
   ```

   som skapar ett objekt som motsvarar intervallet [`x1`, `x2`].

   Vidare ska klassen ha metoderna

   - `public void intersectWith(Interval i)` som beräknar snittet av mängden som aktuellt objekt representerar och mängden som representeras av `i`. Aktuellt objekt ska efteråt innehålla snittet, medan `i` ska vara oförändrad. Snittet av två mängder innehåller alla element som finns i båda mängderna, alltså den delmängd som motsvarar överlappet mellan mängderna.
   - `public boolean equals(Object obj)` från `Object` som avgör om aktuellt objekt (intervall) och (intervallet) `obj` representerar samma mängd av tal. Om `obj` är `null` eller inte är en instans av typen `Interval` så ska metoden returnera `false`.
   - `public String toString()` från `Object` som för icke tomma interval $[x, y]$ (där $x \leq y$) returnerar strängen "[x, y]". För den tomma mängden ska metoden returnera strängen `"empty"`.

   (12p)

6. Implementera metoden

   ```
   public static int[] merge(int[] a, int[] b)
   ```

   som, givet att `a` och `b` är sorterade i stigande ordning, returnerar en (i stigande ordning) sorterad array med alla element i både `a` och `b`. Om t.ex. `a` är `{1,4,8,10}` och `b` är `{2,6,8,9}` så ska metoden returnera arrayen `{1,2,4,6,8,8,9,10}`. Innehållet i `a` och `b` ska inte ändras av metoden.

   (10p)