Tentamen 2001-05-12
Objekt-orienterad programmering i Java, 5p (distanskurs)

Uppsala Universitet
Instutitionen för informationsteknologi
Avdelningen för datalogi

Kursansvarig: Sven-Olof Nyström
Assistenter: Robert Ottenhag, Antonio Addario

11 maj 2001

Läs igenom följande instruktioner noggrant:

- Skrivtiden är fem timmar.
- Inga extra hjälpmedel är tillåtna.
- Skriv endast en lösning per blad och skriv namn på varje blad.
- Skriv inte på baksidan av bladen.
- Hänvisa inte mellan uppgifterna.
- Använd inte rödpenna.
- Om tvetydigheter uppstår, gör intelligenta antaganden och ange dem.
- Oläsbara eller otydliga lösningar ger noll poäng per uppgift.

Maxpoäng är 40. För godkänd krävs vanligtvis 20 poäng och för väl godkänd 30 poäng.

Läs igenom hela tentan innan du börjar. Om du fastnar på en uppgift, gå vidare till nästa—uppgifterna är inte nödvändigtvis ordnade i svårighetsgrad. Om en uppgift består av flera deluppgifter och du fastnar på en av dem, gå vidare till nästa deluppgift—det kan hända att efterföljande deluppgifter kan lösas utan det fullständiga svaret på tidigare uppgifter.

1. **Studiebesök på ICA (5p)**

   Ett äpple kostar 5 kronor och en apelsin 10 kronor. Både äpplen och apelsiner är frukter. En påse kan innehålla 20 frukter. Givet detta:

   (a) Skriv klasserna Frukt, Äpple och Apelsin. Varje frukt ska ha en metod `int pris()` som returnerar fruktens pris. Du får inte använda villkorssatser eller fält/attribut/instansvariabler.

   (b) Skriv klassen Påse. Påse ska definiera följande metoder:
   - `void addera(Frukt f)`. Lägger en frukt `f` i påsen.
   - `int summera()`. Beräknar vad innehållet i påsen totalt kostar.

2. **Undantag (exceptions) (5p)**

   (a) Hur definieras nya typer av undantag i Java?

   (b) Vilka primitiver finns i Java för att hantera undantag och vad betyder de? Ge gärna exampel.

   (c) Vad betyder `throws` i Java? Hur använder man det?

   (d) Ge exempel på någon typ av undantag som definieras i Javas standardbibliotek och när den uppträder.

3. **Cyklar och bilar (5p)**

   Definiera en klasshierarki med följande klasser: Bil, Buss, Fordon, Motorfordon, Cykel. Tips: bilar och bussar är motorfordon. Motorfordon och cyklar är olika typer av fordon.

   Varje klass (utom de abstrakta) ska ha en konstruktor. Du får själv avgöra om klasserna Fordon och Motorfordon ska vara abstrakta, men ange ditt val och ge en motivering!

   Alla fordon med motor ska ha ett fält som anger motorstyrka (antal hästkrafter). Alla fordon ska ha ett fält som anger antalet passagerarplatser. Konstruktorerna ska ta ett argument som anger motorstyrka (i de fall som fordonet har en motor) och ett argument som anger antalet passagerare.

   Det ska vara möjligt att definiera en metod som tar ett antal fordon och beräknar det totala antalet passagerarplatser, eller en metod som tar ett antal motorfordon och beräknar den totala motorstyrkan.

4. **Udda tester (5p)**

Betrakta följande klassdefinition:

```
class Udda {
    public boolean testa(int x) {
        return x%2 == 1;
    }
}
```

Klassen definierar en metod som tar ett heltal som argument och returnerar `true` om argumentet är udda och `false` annars. Givet en variabel `udda` som initierats med `Udda udda = new Udda();` kommer `udda.testa(42)` att returnera `false` och `udda.testa(13)` att returnera `true`.

Här är en annan klass:

```
class Mindre {
    int x;
    Mindre (int x0) {
        x = x0;
    }
    public boolean testa(int y) {
        return y<x;
    }
}
```

Konstruktorn till klassen 'Mindre' tar ett heltalsargument. Givet variabeln `m` som inititierats med `Mindre m = new Mindre(24)` kommer uttrycket `m.test(x)` att returnera `true` exakt i de fall som `x` är mindre än 24.

(a) Definiera ett interface Heltalstest som innehåller metoden `testa`. Visa hur klasserna 'Udda' och 'Mindre' kan skrivas om så att de implementerar interfacet Heltalstest.

(b) Definiera en klass med en metod som tar en samling av heltal och ett heltalstest och returnerar en ny samling bestående av de heltal för vilka testet lyckades (returnerade `true`).
Du får själv bestämma hur samlingarna som ges som argument och returneras som resultat ska representeras. Du kan använda någon av Array, Vector, List, ArrayList, eller LinkedList.

(c) Definiera en ny klass 'Och' som implementerar interfacet Heltalstest. Konstruktorn till klassen 'Och' tar två heltalstest som argument. Heltalstestet 'Och' ska returnera true endast endast i de fall då båda dess argument returnerar true.
Visa hur man kan skapa ett heltalstest som testar om heltalet är mindre än 24 och udda.

5. **Arv, interface, polymorfi (5p)**

(a) Hur kan arv användas vid återanvändning av kod?

(b) I vilka sammanhang kan en subklass referera till sin superklass med nyckelordet `super`? När är det nödvändigt? Ge exempel.

(c) Förklara begreppen *is-a* och `has-a` och hur de relateras till arv och sammansättning. Ge exempel på båda.

(d) Vad menas med att ett program är modulärt? Ge några kriterier.

6. I skogen (5p)

"Overriding" är den viktiga mekanism som låter en subklass ersätta en ärvd generell metod med ett specialfall för den egna klassen.

(a) Vilken text kommer följande program att generera när man kör metoden `main` i klassen `Ramel`?

(b) Klassen `B` ärver från klassen `A`. På samma sätt ärver klassen `C` från klassen `B`. Är `B` en subtyp till `A`? Är `C` en subtyp till klassen `B`? Motivera!

```
class A {
  public String von(String x) { return "Av " + x; }
  public String ist(String x) { return x + " Är"; }
  public String voll(String x, String y) {
    return x + " Full " + y;
  }
}

class B extends A {
  public String ist(String x) { return "Är " + x; }
}

class C extends B {
  public String voll(String x, String y) {
    return y + " " + x + " Full";
  }
}

class Ramel {
  public static void main(String [] args) {
    A a = new A();
    B b = new B();
    C c = new C();
    System.out.println(full(a, a, a) + "!");
    System.out.println(full(a, b, c) + "?");
    System.out.println(full(c, c, c) + ".");
  }
  public static String full(A x, A y, A z) {
    return x.voll(skogen(y), lingonben(z));
  }
  public static String skogen(A x) {
    return x.ist("Skogen");
  }
  public static String lingonben(A x) {
    return x.von("Lingonben");
  }
}
```

7. **Baklängesord (5p)**

Vi vill hitta ord som betyder nåt när man skriver dem baklänges. Till exempel:

amok	koma
atlas	salta
adel	leda
trams	smart
tillit	tillit

Anta att vi har tillgång till en fil med svenska ord.

(a) Definiera en lämplig datastruktur för att lagra ordlistan. (Du får anta att metoder för att läsa in ordlistan från fil finns definierade.)

(b) Skiv en metod som letar upp par av ord som är varandras spegelbild (enligt ovan) och skriver ut dem.

8. **GUI (5p)**

Föreställ dig att du ska implementera ett program med ett grafiskt användargränssnitt (med hjälp av AWT eller Swing).

Programmets gränssnitt ska bestå av ett fönster som visar information i textform och har några knappar för att ta emot kommandon från användaren.

Bland relevanta klasser, interface och metoder i AWT finns `Frame`, `ActionListener`, `actionPerformed`, `paint`, `drawString` och `Graphics`.

Beskriv i stora drag hur programmet ska organiseras.

Vad händer när användaren klickar på en knapp? Vad händer när programmets fönster döljs bakom ett annat fönster för att sedan bli synligt?