Skolan för datavetenskap och kommunikation

Objektorienterad Programkonstruktion, DD1346

Tentamen 2015–06–13, kl. 9.00-12.00

Tillåtna hjälpmedel: Papper, penna och radergummi.

Notera: Frågorna i del I ska besvaras på för ändamålet lämnad plats i tentamenslydelsen. Frågorna i del II besvaras på separat papper – behandla högst en uppgift per sida. Kom ihåg att skriva namn och personnummer på alla inlämnade blad. Skriv tydligt!

Betygsgränser: Betyg FX: $\geq$ 17p i del I
Betyg E: $\geq$ 20p i del I
Betyg D: $\geq$ 20p i del I **och** $\geq$ 5p i del II
Betyg C: $\geq$ 20p i del I **och** $\geq$ 10p i del II
Betyg B: $\geq$ 20p i del I **och** $\geq$ 15p i del II
Betyg A: $\geq$ 20p i del I **och** $\geq$ 20p i del II

Ansvarig: Christian Smith (ccs@kth.se)

Lycka till!

Del I - flervalsfrågor

1. Nedan följer 8 st beskrivningar av programmeringsproblem. För varje problem, ange det designmönster från listan under problemet som är bäst lämpat att lösa det. Varje problem som försetts med rätt mönster ger 1 p. (8 p)

   A) Du har ett flertrådat program där trådinterferens gör att du ibland får oväntade (och felaktiga) resultat från de beräkningar som görs.

   Threadpool **Lock** **Socket** **Protection Proxy**

   B) Ditt program använder ett stort antal i princip identiska objekt, och får problem med otillräckligt minne.

   Flyweight **Threadpool** **Prototype** **Virtual Proxy**

   C) Ditt program blir oacceptabelt långsamt eftersom det interagerar med objekt som finns på en annan dator.

   Publisher/Subscriber **Observer** **Socket** **Remote Proxy**

   D) Ditt program blir oacceptabelt långsamt eftersom det tar så lång tid att skapa många likadana instanser av en viss klass.

   Factory **Clone** **Prototype** **Virtual Proxy**

   E) Du har stora problem att återanvända din kod, eftersom varje klass innehåller fördefinierade referenser till objekt av andra klasser vars metoder den anropar.

   Factory **Observer** **Interface** **Builder**

   F) Din kod går alldeles för långsamt eftersom du har delat upp ett stort problem i för många små problem som alla körs i varsin tråd.

   Flyweight **Parallel Proxy** **Threadpool** **Lock**

G) Din kod är svår att uppgradera i efterhand, eftersom du överallt har definierat explicit vilka klasser som skall instansieras.

Adapter **Prototype** **Observer** **Factory**

H) Ditt program lagrar data i objekt av en viss typ. Programmet går antingen långsamt i uppstarten, när data lagras, eller senare under körningen, när data läses, beroende på vilken typ av objekt du väljer. Det finns ingen specifik datastruktur som ger snabb körning i båda fallen.

Flyweight **Builder** **Prototype** **Factory**

2. I denna uppgift finns 3 kodlistningar. För varje kodlistning, ange om den går att kompilera utan fel[1]. Varje korrekt identifierad kodlistning ger 1 p. (3 p)

A)
```
public class ClassA {

    public final int a = 1;

    public static void main(String[] args){

        System.out.println(a);
    }

}
```

[1]Som referens används kompilatorn javac till Java 7, dvs den som finns som standard i skolans datorsalar

B)
```
public class ClassB {

    private static ClassB myB = new ClassB();
    private int a = 1;

    public static void main(String[] args){

        System.out.println(myB.a);
    }

}
```

C)
```
public class ClassC {

    private static InnerClassC myInnerC = new InnerClassC();
    private int a = 1;

    public static void main(String[] args){

        System.out.println(myInnerC.getA());
    }

    public static class InnerClassC extends ClassC{

        public int getA(){
            return super.a;
        }

    }

}
```

3. För varje påstående nedan om `protected`, ange om det är sant eller falskt. 4 korrekta svar ger 2p, 3 korrekta svar ger 1p, 2 eller färre korrekta svar ger 0 p (2 p)

   A) Nyckelordet `protected` i Java används för att göra skrivskyddade fält vars värden bara går att läsa, men inte ändras.

   B) Inre och nästade klasser är de enda klasser som får deklareras med nyckelordet `protected` i Java.

   C) I Java är `protected` den mest restriktiva åtkomstgraden som man kan ge ett fält eller en metod.

   D) Om ett fält har deklareratas med nyckelordet `protected` så får det endast användas i metoder som också är deklarerade som `protected`.

4. För varje påstående om trådar nedan, ange om det är sant eller falskt. 5 korrekta svar ger 4p, 4 korrekta svar ger 2p, 3 korrekta svar ger 1p, 2 eller färre korrekta svar ger 0 p (4 p)

   A) Trådar har alltid separata minnesareor, och två olika trådar kan inte läsa och skriva samma variabler.

   B) I Java är det tillåtet att skapa fler trådar än vad man har processorkärnor.

   C) Nyckelordet `synchronized` i Java används till att låta trådar vänta på varandra, så att de avslutar samtidigt.

   D) Kommandot `Thread.join()` i Java används för att låta två trådar interagera, genom att lägga ihop deras variabler till samma minnesarea.

   E) **Threadpool** innebär att man skapar ett förutbestämt antal trådar, och låter dessa hämta uppgifter ur en uppgiftskö.

5. För varje påstående om klasser och instanser i Java, ange om det är sant eller falskt. 5 korrekta svar ger 4p, 4 korrekta svar ger 2p, 3 korrekta svar ger 1p, 2 eller färre korrekta svar ger 0 p (4 p)

   a) En instans kan bestå av flera olika klasser.

   b) Nyckelordet `static` används för att deklarera klassvariabler och klassmetoder.

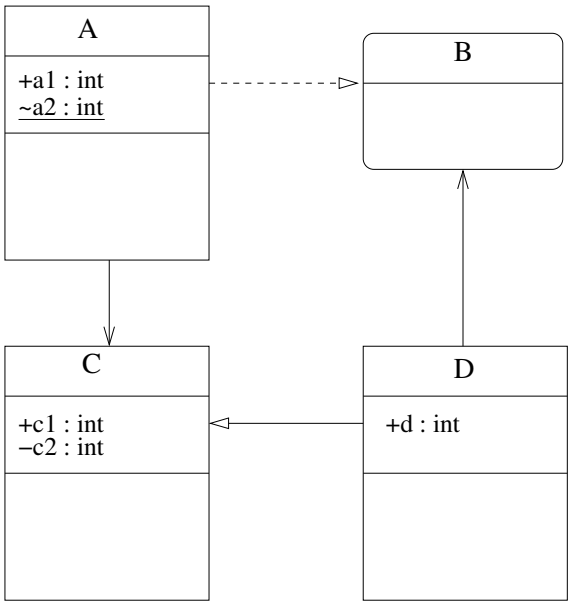
   c) Nyckelordet `abstract` används för att deklarera klasser som inte går att instansiera.

   d) Ett Javaprogram består alltid av minst en klass.

   e) Det är tillåtet att skriva program i Java som inte skapar några instanser av någon av de klasser det består av.

6. Betrakta följande UML-diagram. För varje bit kod som följer nedan, ange i vilken eller vilka klass(er) och/eller gränssnitt det vore rimligt att hitta den, givet diagrammet. Varje korrekt svar består alltså av en kombination av 0 eller fler av namnen {**A**, **B**, **C**, **D**}. Notera att kodbitarna inte behöver vara kompletta rader, utan kan vara en del av en rad, t.ex skulle den hypotetiska kodbiten '`int a`' kunna tänkas vara en bit av raden '`static int a2;`', och då passat in i **A**. Du kan dessutom anta att alla variabelnamn är unika, dvs '`a2`' refererar alltid till fältet i **A**, och inte till någon lokal variabel någon annanstans.

   Varje korrekt svar ger 0.5 p. (4 p)



- `implements C`
- `private B myB;`
- `implements B`
- `import D.*;`
- `a1++`
- `c1++`
- `c2 += d;`
- `extends C`

Del II - fördjupningsfrågor

Följande uppgifter besvaras på separat papper.

7. A) Vad menas med *lös koppling*? Varför är det önskvärt? (2 p)

   B) Vad menas med *polymorfism*? Hur kan det underlätta när man vill åstadkomma lös koppling? (2 p)

   C) Hur kan åtkomsttypen för fält och metoder användas för att underlätta när man vill åstadkomma lös koppling? (2 p)

8. Förklara designmönstret *Composite*. Vad är dess syfte, hur implementerar man det, och vilka fördelar har det (eller nackdelar, om dessa är mer relevanta)? (3 p)

9. En vän ber om hjälp med lite felsökning. Hen skrivit ett program för att skicka enklare textmeddelanden mellan två datorer (ungefär som i projektuppgiften i denna kurs). Man har en ruta **A** (JTextArea) där man kan skriva in den text som man själv vill skicka. Man har också en separat tråd med en loop som läser vad ens samtalpartner skrivit, från en socket som är ansluten till partnerns dator. Både den text man själv skrivit och den text som partnern skrivs visas i en scrollbar ruta **B** (en JLabel inom en JScrollPane). Alla nya meddelanden läggs till i slutet på texten i **B**, så att man kan scrolla upp och se hela konversationen.

   För det mesta fungerar allt som man vill, men ibland (ytterst sällan) händer det att antingen ett meddelande man själv skrivit eller ett som partnern har skrivit inte läggs till i **B**, utan att programmet uppvisar några andra fel i övrigt. Vad har vännen troligtvis gjort för fel, och hur kan hen enklast åtgärda det? (3 p)

10. Som ett sommarjobb har du fått i uppgift att skriva ett Java-program för bildbearbetning. Programmet ska fungera som en server, till vilken andra program kan ansluta. De andra programmen ska kunna skicka en bildfil och instruktioner om vad som ska göras med bilden, t.ex att ändra upplösning, byta filformat, byta ut alla hundar i bilden mot katter, mm. Din server ska sedan skicka tillbaka en ny bildfil, som har bearbetats enligt önskemålen.

    För att spara tid har du letat på internet efter fri programvara, och hittat ett tiotal olika Java-program som tillsammans kan utföra alla de bildbehandlingsuppgifter som efterfrågas. Dessa program är skrivna av olika personer, håller olika kvalitet, och använder olika API:n. Vissa av dem underhålls väl och kommer regelbundet ut med uppdaterade versioner, medan andra eventuellt är övergivna och kan komma att behöva ersättas med nya program så småningom.

    Din uppgift är till att börja med att skriva den första versionen av programmet, men eftersom du bara är tillfälligt anställd över sommaren vet du inte vem som ska driva det vidare till hösten.

Beskriv hur du strukturerar din server. Hur fungerar dess olika delar, och hur kommunicerar de med varandra. Namnge korrekt de designmönster som ingår i strukturen. Motivera varför din lösning är bra. (7 p)

11. Medan många designmönster har som främsta ändamål att göra det lättare för programmeraren, genom att t.ex underlätta kodåteranvändning eller göra det lättare att förstå hur olika klasser ska användas, syftar andra mönster främst till att förbättra det färdiga programmets egenskaper, så att det kan köras effektivare och med färre exekveringsfel. Ange tre designmönster som har som syfte att förbättra programmets körning, och förklara hur. (6 p)