Skolan för datavetenskap och kommunikation

Objektorienterad Programkonstruktion, DD1346

Tentamen 2012–03–13, kl. 09.00-12.00
Inklusive lösningsförslag

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. Använd gärna både fram- och baksida, men behandla högst en uppgift per sida. Kom ihåg att skriva namn och personnummer på alla inlämnade blad. Skriv tydligt!

Betygsgränser: Betyg XF: $\geq$ 18p i del I
Betyg E: $\geq$ 20p i del I
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. Punkterna a) till h) nedan beskriver olika designmönster. För varje beskrivning, ange namnet på det mönster som bäst passar beskrivningen. Välj namn från listan nedan. Notera att det finns fler namn än beskrivningar - alltså behöver inte alla namn användas. Varje rätt svar ger 1 poäng; totalt kan denna uppgift alltså ge 8 poäng.

Singleton	**MVC**	**Composite**	**Proxy**	**Adapter**	**Flyweight**
Lock	**Observer**	**Factory**	**Builder**	**Prototype**	**Facade**

   a) Man skapar nya objekt **O** genom att skapa kopior av ett objekt **A** som redan existerar. Prototype (1 p)

   b) Man skapar nya objekt **O** med anrop av en metod **B**() som beslutar vilken typ av objekt som skall returneras. Factory (1 p)

   c) Man skapar nya objekt **O** genom att först skapa ett objekt **A**. Objektet **A** kan man sedan modifiera och bygga på tills man är nöjd. Till sist låter man **A** returnera objektet **O**. Builder (1 p)

   d) Objektet **O** är den enda instansen av klassen **A** som kan existera. Singleton(1 p)

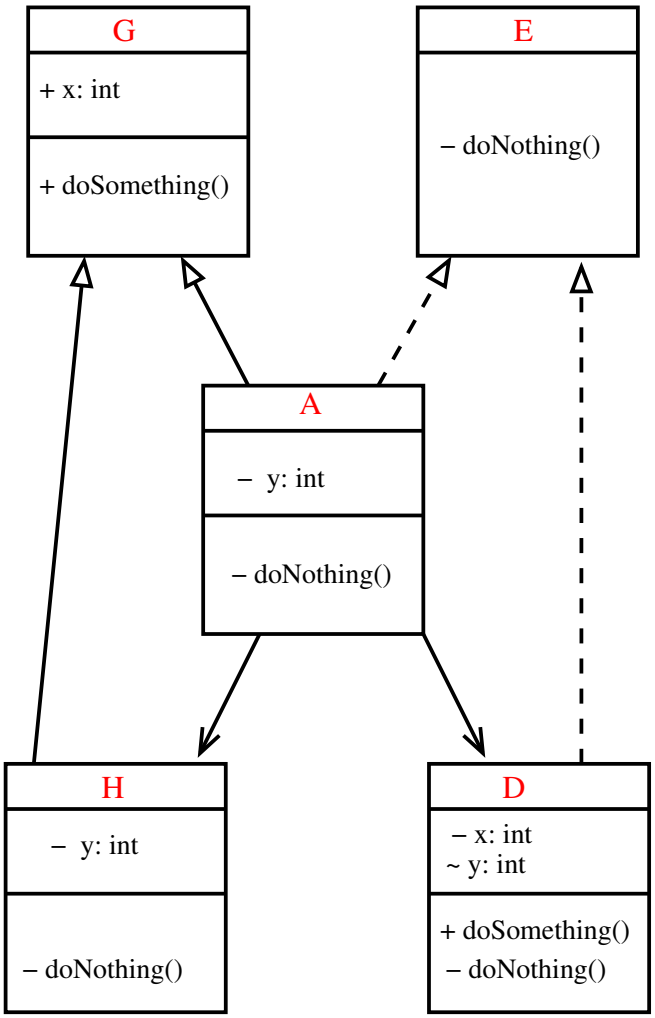
   e) Man skapar ett objekt **O** som möjliggör för de annars inkompatibla objekten **A** och **B** att interagera. Adapter (1 p)

   f) Man skapar ett objekt **O** som ger ett enkelt högnivå-API till en komplicerad samling av lågnivåklasser. Facade (1 p)

   g) Man har ett objekt **O** som bevakar ett annat objekt **A**, och utför något när **A**:s tillstånd ändras. Observer (1 p)

   h) Man skapar en större mängd objekt, där man sparar resurser genom att lagra gemensamma data centralt, utanför objekten. Flyweight (1 p)

2. Fälten för namnen på delarna i nedanstående klassdiagram är tomma. Fyll i fälten med rätt namn (A–J) från klass- och gränssnittsdefinitionerna som följer från nästa sida. Vissa definitioner kan beskriva klasser/gränssnitt som inte fungerar eller inte finns med i UML-diagrammet. Det finns totalt 10 definitioner, från vilka 5 namn skall väljas ut. Varje rätt ifyllt namnfält ger 1 p. (5 p)



```
public class A extends G implements E{

    private int y;

    private void doNothing(){
       D myD = new D();
       H myH = new H();
    }
}


public class B extends A implements E{

    private int x;
    protected int y;

    public void doSomething(){
    }

    private void doNothing(){
    }
}

public class C {

    private int x;

    private void doSomething(){
    }
}

public class D implements E{

    private int x;
            int y;

    public void doSomething(){
    }

    private void doNothing(){
    }
}
```

```
public interface E {

    private void doNothing();
}

public interface F extends E {

    private int x;
            int y;

    public void doSomething();

    private void doNothing();
}

public class G {

    public int x;

    public void doSomething(){
    }
}

public class H extends G {

    private int y;

    private void doNothing(){
    }

}

public class I extends C implements E{

    int y;

    private void doNothing(){

        H myH = new H();

        F myF = new F();
    }

}
```

```
public interface J extends C implements A{

    private int y;

    private void doNothing(){
    }

}
```

3. I denna uppgift följer ett exempelprogram. För varje fråga, ange korrekt svar.

```
public class Kod_2a {

    public static void main(String[] args){

        int i = 0;
        int j = i;
        int k = 0;

        while( i++ < 10 ){
            k = i;
        }

        System.out.println(i + ":" + j + ":" + k);
    }
}
```

a) Vilket av alternativen nedan beskriver den korrekta utskriften från programmet `Kod_2a`? (1 p)

(i) `9:0:9`
(ii) `9:9:9`
(iii) `10:0:9`
(iv) `10:0:10`
(v) `10:10:9`
(vi) `11:0:10`
(vii) `11:11:10`
(viii) Utskriften kan bli olika från gång till gång, men `i` får alltid samma värde som `j`
(ix) Utskriften kan bli olika från gång till gång, men `i` får alltid samma värde som `k`

b) Vilket (om något) av alternativen *i*–*ix* ovan beskriver den korrekta utskriften från programmet `Kod_2a`, om man skulle ändra villkoret i while-loopen till `( ++i < 10 )`? iii (1 p)

c) I vilket av fallen ovan körs loopen flest gånger? Markera rätt alternativ (1 p)

(i) Den ursprungliga koden
(ii) Koden enligt uppg b
(iii) Lika många gånger i båda

4. Antag att klasserna `X` och `Y` ingår paketet, `XY`, och är definierade i varsin fil:

```
package XY;
public class X{
    private    int a = 1;
    static     int b = 2;
    protected  int c = 3;
    public     int d = 4;

    public int x(){
        return a;
    }

    public static int y(){
        return b;
    }

}

package XY;
public class Y extends X{

    public int x(){
        return c;
    }

    public static int y(){
        return 2*b;
    }
}
```

a) Antag att man skapar en klass Z som också ligger i paketet XY och som ärver från Y. Antag att man i Z inte överskuggar några av de ärvda fälten. Vilka av fälten a, b, c och d kommer då att vara direkt åtkomliga, t.ex med anrop av typen `a = 13;` i Z? b,c,d (1 p)

b) Antag att man skriver en klass K som inte ligger i samma katalog eller paket som X och Y, men som importerar paketet XY. I K skapar man sedan ett objekt av typen X med följande rad:
`X myX = new X();`
vilka av fälten a, b, c och d kan man nu komma åt direkt med `myX`, dvs med anrop av typen `myX.a = 2;`? d (1 p)

c) Antag att man i klassen K enligt ovan skapar ett objekt med följande rad:

```
X myX = new Y();
```
Vilket heltal kommer att returneras av ett anrop av `myX.x()`? 3 (1 p)

c) Antag att man i klassen K enligt ovan anropar `myX.y()`. Vilket heltal kommer att returneras? 2 (1 p)

5. För varje påstående om sockets i Java, 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) Det går bara att ha en öppen `ServerSocket` åt gången. Falskt

b) En `ServerSocket` och en `Socket` fungerar likadant, med undantaget att en `ServerSocket` kan vara ansluten till flera andra motparter. Falskt

c) Det går att använda `Socket` för att ansluta två program på samma dator till varandra. Sant

d) Om man har kopplat ihop två program, på två olika datorer anslutna till samma nätverk, med `Socket` och skickar meddelanden med TCP/IP, så är man garanterad att inga meddelanden kommer fram utan att vara i samma kronologiska ordning som de skickades. Sant

6. Vilka av följande påståenden angående arv i Java är korrekta? Skriv "sant" eller "falskt" för varje påstående. 5 korrekta svar ger 4 p, 4 korrekta svar ger 2p, 3 korrekta svar ger 1p, 2 eller färre korrekta svar ger 0 p (4 p)

a) Ett gränssnitt (interface) kan ärva från flera andra gränssnitt. Sant

b) En abstrakt klass kan ärva från flera andra abstrakta klasser. Falskt

c) Om (den icke-abstrakta) klassen **A** ärver från (den icke-abstrakta) klassen **B**, så kommer ett anrop av en argumentlös konstruktor i **A** alltid att anropa den argumentlösa konstruktorn i **B**. Sant

d) Om (den icke-abstrakta) klassen **A** ärver från (den icke-abstrakta) klassen **B**, så måste **A** ha precis lika många fält som **B**. Falskt

e) Man kan deklarera klasser så att de inte ärver från klassen Object. Falskt

Kommentar:

strikt sett borde 6c ha förtydligats till:

"Om (den icke-abstrakta) klassen **A** ärver från (den icke-abstrakta) klassen **B**, så kommer ett anrop av en argumentlös konstruktor i **A** alltid att anropa den argumentlösa konstruktorn i **B**, om **A**:s argumentlösa konstruktor inte explicit anropar en annan konstruktor i **B** ."

Det här felet uppmärksammades när den kursomgången gick, och vi beslöt att tillåta båda svarsalternativen.

Del II - fördjupningsfrågor

Följande uppgifter besvaras på separat papper.

7. Vad är skillnaden mellan en grund kopia och en djup kopia? Du kan svara med ord, UML-diagram eller både och, så länge svaret är tydligt och entydigt. (2 p)

   I en GRUND KOPIA kopieras alla fält precis som de står, dvs, om ett fält pekar på objektet A i originalet O, så kommer motsvarande fält att peka på samma objekt A i kopian K. I en DJUP KOPIA så kopierar man även de objekt som refereras till, dvs, om ett fält pekar på objektet A i originalet O, så skapar man ett nytt objekt B, som är en kopia av A, och låter det motsvarande fältet i K peka på B.

8. a) Vilka två huvudsakliga sätt finns det att explicit skapa separata trådar i Java? (1 p)

   Sätt 1: Ärv av klassen Thread, och skriv det du vill att tråden ska utföra i dess run()-metod. Sätt 2: skapa ett objekt som implementerar gränssnittet Runnable, skriv det du vill att tråden ska utföra i dess run()-metod, och skapa ett Thread-objekt med din Runnable som argument i konstruktorn.

   b) Förklara för- respektive nackdelar med de olika sätten. (2 p)
   Fördelen med sätt 1 ovan är att det ofta blir mindre kod att skriva. Nackdelen är att man inte kan ärva från någon annan klass än Thread. Fördelen med sätt två ovan är att man kan ärva från vilken klass man vill, samt att man är mer flexibel i vilka mönster man kan använda, t.ex threadpool. Nackdelen är att det ofta blir mer kod att skriva (och läsa/överblicka).

9. Antag att du ska skapa en ny klass **A**, så att objekt av typen **A** kan innehålla godtyckligt långa sekvenser av flyttal. Antag att du ämnar använda **A** så att du stoppar in tal i den vid ett antal olika tillfällen i ett programs uppstartsfas, för att sedan läsa dess innehåll vid ett stort antal upprepade tillfällen under programmets körning.
   Du förväntar dig att **A** kan komma att innehålla ett ganska stort antal tal, och att hur många som ska stoppas in i vilken del av uppstartsfasen kommer att bero på yttre faktorer du inte kan påverka. Du vill förstås att din kod ska vara så effektiv som möjligt, både vad gäller minnesanvändning och exekveringstid.

   a) Namnge ett lämpligt designmönster att använda. (1 p)
   Här passar mönstret Builder bra, eftersom det tillåter olika strukturer vid konstruktion resp användning av ett objekt

   b) Beskriv konkret hur man skulle kunna implementera detta mönster för **A**. Du behöver inte skriva någon kod. (2 p)
   Vi skapar en Builder-klass som använder en datastruktur som tillåter godtyckliga tillägg (t.ex en länkad lista) i uppbyggnadsfasen. När vi är färdiga med att

bygga vår datastruktur så kan vi konvertera den till en struktur som är effektivare för lagring och läsning (t.ex en array) och returnera ett objekt av typen A som innehåller datan i strukturen som är optimerad för lagring/läsning

c) Antag att du vill kunna loopa igenom alla talen i **A** med en *for-each*-loop. Vilket designmönster använder du lämpligen då, och hur implementeras detta i Java? (2 p)

Vi använder oss av mönstret Iterator. I Java gör vi detta genom att vi låter A implementera gränssnittet Iterable, och alltså kunna returnera ett Iterator-objekt som kan returnera alla de lagrade flyttalen i ordning

10. Antag att man har 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 är det troligen som går fel, och hur kan man enkelt åtgärda det? Du bör svara i konkreta termer, men behöver inte skriva någon kod. (5 p)

Eftersom vi har två separata trådar som kan anropa metoden som lägger till text sist i textfönstret B, så är det troligt att det iblan händer att båda trådarna anropar denna metod samtidigt. Dvs, det händer ibland att både den lokala användaren och dennes motpart skriver något samtidigt. Då kommer båda trådarna att läsa in texten i B innan några tillägg gjorts, lägga till sitt respektive tillägg och sedan skriva den nya texten med (enbart) det egna tillägget. Enbart texten från den tråd som avslutar skrivandet sist kommer att läggas till i B.

Detta kan åtgärdas genom att vi trådsäkrar metoden för att lägga till ny text i B, genom att använda nyckelordet synchronized i metoddeklarationen. Då kommer bara en tråd i taget att tillåtas lägga till ny text, och en av trådarna får vänta tills den andra är helt färdig med sitt tillägg innan den tillåts göra sitt eget tillägg.

11. Antag att programmet i föregående uppgift har vuxit med ett stort antal tillagda funktioner. Numera kan man skicka filer, koppla upp sig mot flera användare, chatta med både ljud och bild samt spela små enkla multiplayerspel. Antag att du har skapat en klass **A** som kan hantera alla olika inställningar till ditt program. **A** innehåller ett stort antal privata fält där man lagrar saker som portnummer, användarnamn, tidskonstanter för hur länge programmet ska vänta innan det ger upp, inställningar för din mikrofon och webkamera, sökvägar till loggfiler, och många andra inställningar för hur programmet ska se ut och bete sig. **A** har ett stort antal publika metoder för att hämta eller ändra värdena på dessa fält (getters/setters).

    a) Ange ett (eller flera) lämpligt/lämpliga designmönster att använda vid skapandet av **A**. Motivera ditt svar! (2 p)
    A bör vara en Singleton, eftersom man inte vill ha flera konkurrerande uppsättningar med inställningar. Man kan med fördel använda en fabriksmetod för att skapa singletonobjektet. Dessutom kan man tänka sig att man använder sig av en virtuell proxy, om man förväntar sig att det finns inställningar som är svåra att skapa (t.ex för kameran).

    b) Ange ett lämpligt designmönster för att strukturera kommunikationen mellan **A** och resten av programmet. Motivera ditt val och förklara hur mönstret fungerar i detta sammanhang! (2 p)
    Observer är lämpligt att använda här. I detta sammanhang låter man alla objekt som beror på inställningarna lyssna på A, och A meddelar dessa om någon inställning ändras.

    c) Antag att man vill att alla inställningar automatiskt ska sparas i en fil, som automatiskt läses in nästa gång man startar programmet. Förklara hur detta kan göras, och hur/var dessa funktioner passar in i mönstren du angett som svar i deluppgift a och b ovan! (3 p)
    Eftersom vi vet att inställningarna kommer att behövas när programmet skapas, kan vi låta A läsa in alla värden direkt vid uppstart, dvs vi skapar A redan innan A efterfrågas första gången. Om vi har valt att använda en virtuell proxy läser vi lämpligen alla de nödvändiga värdena direkt, och skapar vår proxy, för att sedan bara läsa in övriga värden (de som kräver mer resurser/tid för att skapa) när de behövs. Vad gäller sparandet av inställningar, kan det vara onödigt att skriva till fil varje gång något enskilt värde ändras (Med sliders och andra grafiska interface kan det ju bli väldigt många ändringar). I stället låter vi en inställningssparare lyssna på om något ändras, och i sådana fall spara ner värden när programmet avslutas.
    Om vi har väldigt mycket data att spara, som bilder eller andra större datamängder, kan vi med fördel dela up den sparade informationen på flera separata filer, och bara spara de bitar som faktiskt har ändrats. Vid inläsning i uppstarten kan man då nöja sig med att läsa in en huvudfil med de nödvändigaste inställningarna direkt, för att vänta med annan data tills den efterfrågas för

första gången (virtuell proxy).

12. a) Är det möjligt att skapa en klass i Java som varken kan instansieras eller ärvas ifrån? (1 p) Ja

    b) Skulle en sådan klass vara meningsfull? Förklara varför/varför inte! (2 p) En sådan klass kan vara meningsfull. Den kan innehålla statiska metoder, och till exempel fungera som ett matematikbibliotek eller en fabrik som skapar andra objekt.