Skolan för datavetenskap och kommunikation

Objektorienterad Programkonstruktion, DD1346 FACIT

Tentamen 2016–03–18, kl. 14.00–17.00

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. I denna uppgift följer beskrivningar av 6 olika problem du kan träffa på under programutveckling. För varje problem, ange vilket designmönster från listan nedan som är bäst lämpat att lösa det. Varje korrekt angett mönster ger 1 p. (6 p)

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

   a) Ditt program skapar väldigt många objekt som är till stora delar identiska och som tar upp onödigt mycket minnesutrymme. Flyweight

   b) Ditt program skapar väldigt många objekt som åtminstone till en början är i princip identiska, och som behöver väldigt resurskrävande beräkningar för att skapas. Prototype

   c) Du ska implementera en ny klass, men tvingas välja om du skall använda en intern datastruktur som är för otymplig för att kunna skapa nya objekt effektivt, eller en som gör att det går väldigt långsamt att använda de färdiga objekten. Builder

   d) Ditt program har flera trådar som samtidigt använder samma resurser och orsakar trådinterferens. Lock

   e) Du håller på att utveckla ett program som kommer att ha mycket interaktion med användare, men du tvivlar på att ditt nuvarande GUI är bra som det är, utan tror att man kommer att vilja byta ut det mot ett annat vid ett senare tillfälle. MVC

   f) Du har precis hittat på en helt ny datastruktur för att effektivt lagra en viss typ av objekt. Tyvärr fungerar inte din nya datalagringsklass med befintliga klassers metoder för att stega sig igenom alla de objekt den innehåller. Iterator

2. I denna uppgift finns 3 kodlistningar. För varje kodlistning, ange vilket mönster (från listan nedan) som koden beskriver. *Observera att generiska namn används i stället för standardiserade namn, samt att koden kan vara förenklad av utrymmesskäl, och går inte nödvändigtvis att kompilera.* Varje korrekt namngiven kodlistning ger 1 p. (3 p)

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

a)

Factory

```
public class A{

    public static GenericType getObject(int a){
        if(decisionFunction(a)){
            return new SpecificType1();
        }else{
            return new SpecificType2();
        }
    }
}
```

b)

Lock

```
public class B{

    private int a = 0;

    public synchronized void setA(int newA){
        a = newA;
    }

    public synchronized int getA(){
        return a;
    }

    public synchronized void incA(int b){
        a += b;
    }

}
```

c)

Flyweight

```
public class C{

    private static final HeavyType myHeavy = new HeavyType();
    private int a;

    public C(int setA){
        a = setA;
    }

}
```

3. Här följer 5 st kodlistningar i Java[1]. För varje kodlistning, ange om den kommer att fungera, ge kompileringsfel eller ge fel när man kör den. Kod anses fungera **omm** den ger en deterministisk utskrift av ett tal vid körning. Anta att varje klass **X** finns i en fil som heter '`X.java`', och kompileras med kommandot '`javac X.java`', och körs med '`java X`'. (5 p)

a)

Fungerar

```
public class A{

    public static void main(String[] args){
        A myA = new A();
    }

    private A(){
        this(5);
    }

    private A(int a){
        System.out.println(a);
    }

}
```

b)

Kompilerar EJ

```
public class B{

    B myB;

    public static void main(String[] args){
        myB = new B(5);
    }

    private B(int b){
        System.out.println(b);
    }

}
```

[1]För tydlighets skull anses här Java 8, som finns i skolans datasalar

c)

Fungerar

```
public class C{

    public static void main(String[] args){

        C myC = new C(){
            public void printC(int c){
                System.out.println(c);
            }
        };

        myC.printC(5);
    }

    public void printC(int c){
        System.out.println(c * Math.random());
    }

}
```

d)

Ger icke-determinstiskt resultat

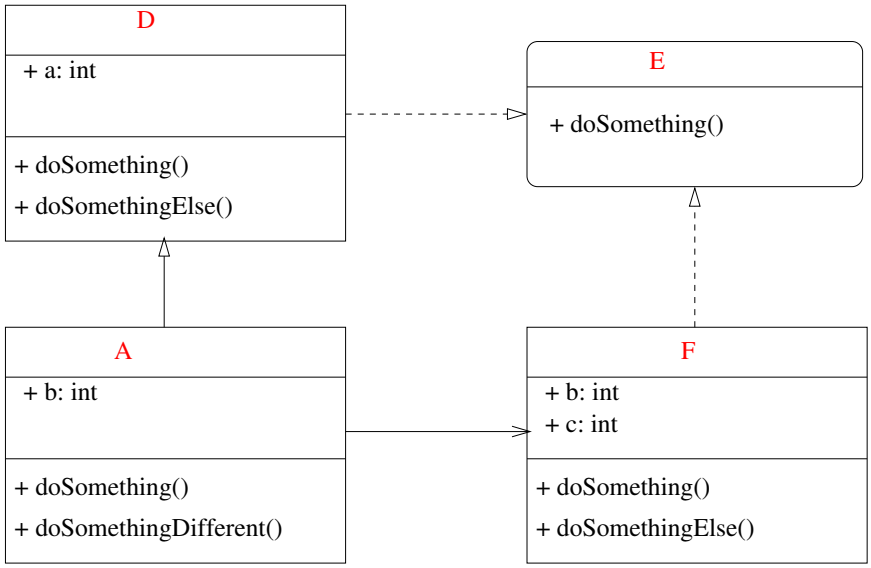
```
public class D implements Runnable{

    static int d = 0;

    public static void main(String[] args){
        Thread th1 = new Thread(new D());
        Thread th2 = new Thread(new D());
        Thread th3 = new Thread(new D());

        th1.start();
        th2.start();
        th3.start();

        System.out.println(d);

    }

    public void run(){
        while(d < 50000){
            incD();
        }
    }

    private synchronized void incD(){
        d++;
    }

}
```

e)

Ger icke-deterministikt resultat

```
public class E extends Thread{

    static int e = 0;
    static Object lock = new Object();

    public static void main(String[] args){
        E E1 = new E();
        E E2 = new E();
        E E3 = new E();

        E1.start();
        E2.start();
        E3.start();

        System.out.println(e);

    }

    public void run(){
        doStuff();
    }

    static void doStuff(){
        synchronized(lock){
            while(e < 50000){
                e++;
            }
        }
    }
}
```

4. Nedan följer ett UML-diagram och ett antal olika kodlistningar. För varje ruta i UML-diagrammet, ange (med en bokstav A-H) vilket namn den borde ha, utifrån vilken av kodlistningarna som bäst beskriver den. Alla rutorna korrekt namngivna ger 3 poäng, tre korrekt namngivna ger 2 poäng, och två korrekt namngivna ger 1 poäng. (3 p)



A)

```
public class A extends D{

    public int b = 3;
    private int c = 3;
    private F myF;

    public void doSomething(){
        myF = new F();
        myF.doSomthingElse();
    }

    public void doSomethingDifferent(){
        b -= c;
    }


}
```

B)

```
public interface B extends A{

    public int a = 3;

    public void doSomething(){
        a++;
    }

    public void doSomethingElse(){
        a--;
    }

}
```

C)

```
public class C implements D{

    public int b = 3;

    public void doSomething(){
        a++;
    }

    public void doSomethingDifferent(){
        a--;
    }

}
```

D)

```
public class D implements E{

    public int a = 3;

    public void doSomething(){
        a++;
    }

    public void doSomethingElse(){
        a--;
    }

}
```

E)

```
public interface E{

    public void doSomething();

}
```

F)

```
public class F implements E{

    public int b = 3;
    public int c = 3;

    public void doSomething(){
        b++;
    }

    public void doSomethingElse(){
        c--;
    }

}
```

G)

```
public class G extends F{

    public int b = 3;
    private int c = 3;
    private F myF;

    public void doSomething(){
        myF = new F();
        myF.doSomthingElse();
    }

    public void doSomethingDifferent(){
        b -= c;
    }

}
```

H)

```
public interface H implements D,F{

    public void doSomething();

}
```

I)

```
public class I implements E{

    public int b = 3;
    public int c = 3;

    public void doSomething(){
        b++;
    }

    public void doSomethingElse(){
        c--;
    }

}
```

5. För varje påstående om åtkomst 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) Om ett fält deklareras med nyckelordet `protected` så kan dess innehåll inte ändras efter initiering. Falskt

   b) En metod som deklareras som `protected` kan bara anropas från klassen självt. Falskt

   c) Om en metod deklareras med nyckelordet `private` så kan den inte anropas av en ärvande klass. Sant

   d) Metoden `main()` måste alltid deklareras som `private` för att förhindra att andra klasser anropar den och orsakar trådintereferens. Falskt

6. För varje påstående om polymorfism 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) Om en variabel deklarerats med en superklass **A** kan man instansiera den med en klass **B** som ärver från **A**. Sant

   b) Eftersom det inte går att instansiera abstrakta klasser, får man inte deklarera variabler med abstrakta klasser. Falskt

   c) Det är tillåtet att deklarera variabler med gränssnitt. Sant

   d) Ett gränsnitt kan implementera en klass om den är abstrakt. Falskt

   e) Om en variabel deklarerats med en klass **A** kan man instansiera den med en superklass **B** som **A** ärver ifrån. Falskt

7. För varje påstående om trådar 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) Om man använder en `Threadpool` måste man ha lika många trådar som man har processorkärnor. Falskt

   b) Varje tråd kan ha flera olika programpekare. Om dessa pekar på samma rad i programmet uppstår ett *deadlock*. Falskt

   c) *Trådinterferens* kan förhindras genom att deklarera variabler som `private`. Falskt

   d) I Java kan man starta en ny tråd genom att anropa metoden `start()` på ett objekt av typen `Thread` Sant.

Del II - fördjupningsfrågor

Följande uppgifter besvaras på separat papper.

8. Förklara följande begrepp: **thread interference**, **deadlock** och **starvation**. (3 p)

9. Förklara vad gränssnitt och abstrakta klasser är och hur de används. Hur kan de underlätta för att åstadkomma lös koppling? (3 p)

10. Du har blivit ombedd att skriva program som kräver att man använder två smartphones, i ett försök att få upp försäljningen av dessa. Din uppdragsgivare tänker sig att man kan ha ett tangentbord på den ena telefonen och använda skärmen på den andra för att se texten man skriver. Eftersom uppdragsgivaren är kopplad till en större nätverksleverantör, vill de att all kommunikation mellan telefonerna ska ske över internetanslutningar.

    Till en början räcker det att kunna skriva och redigera texter, men din uppdragsgivare kan säkert komma på ny innovativ funktionalitet i framtiden, vilket du förstås bör ta höjd för. Skriv och förklara hur du strukturerar programmet och hur programmets olika delar kommunicerar med varandra. Namnge alla ingående designmönster korrekt. (7 p)

11. Din vän har precis börjat programmera i Java, men upplever att alla program hen skriver tar väldigt lång tid på sig att starta. Skriv en kort guide till hur man kan använda olika designmönster för att göra sina program snabbare i uppstarten, och namnge alla relevanta mönster korrekt. (5 p)

12. Hur fungerar designmönstrena **Adapter** och **Proxy**? Vilka likheter har de, och vilka skillnader? Hur kan man implementera dem? Rita en UML och förklara! (6 p)