Övningsuppgifter i Objektorienterad modellering och design (OMD) Helsingborg

EDAF25

19 december 2017

Innehåll

1 Objektorienterad modellering **2**

2 Design smells **6**

3 Grundläggande designprinciper (SRP, OCP, DIP, DRY) **8**

4 Designmönster (objekt, sekvens och tillståndsdiagram) **11**
4.1 Kommandomönstret 11
4.2 Kompositmönstret 11
4.3 Mallmetodmönstret 11
4.4 Strategimönstret 11
4.5 Null Objekt 11
4.6 Dekoratörsmönstret 11
4.7 Observatörsmönstret 11
4.8 Tillståndsmönstret 11
4.9 Fabriksmönstret 11
4.10 Alla mönster 11

5 Ytterligare designprinciper (ISP, muterbarhet, LSP) **12**

6 Arkitekturmönster **13**
6.1 MVC 13

7 Paketdesign **14**
7.1 Ramverk 14
7.2 Designprinciper (CRP, REP, CCP, ADP, SDP, SAP) 14

Kapitel 1

Objektorienterad modellering

Uppgift 1.1
Beskriv följande i ett klassdiagram: I ett system för bokning av undervisningssalar har varje sal ett namn och ett antal platser. Det finns salar för tre ändamål: föreläsning, övning och tentamen. För varje bokning anger man salens namn och önskad tid (dag, starttid och sluttid). Varje bokning tilldelas ett bokningsnummer.

Uppgift 1.2
Denna uppgift innehåller ett antal deluppgifter, där varje deluppgift har ett påstående och en anledning. För varje deluppgift, svara med ett av följande alternativ:

A Både påståendet och anledningen är korrekta uttalanden och anledningen förklarar påståendet på ett korrekt sätt.

B Både påståendet och anledningen är korrekta uttalanden, men anledningen förklarar inte påståendet.

C Påståendet är ett korrekt uttalande, men anledningen är ett felaktigt uttalande.

D Påståendet är felaktigt, men anledningen är ett korrekt uttalande.

E Både påståendet och anledningen är felaktiga uttalanden.

	Påstående	Anledning
T1	Den viktigaste egenskapen för mjukvara är att den kan underhållas och återanvändas.	Det är bra att följa goda OO principer från start för att få ett väldesignat projekt.
T2	Med hjälp av OMD kan man mäta kvaliteten på mjukvaran.	OMD handlar om att rita olika typer av diagram.
T3	Ett sätt att åstadkomma inkapsling (encapsulation) är att minimera synligheten för metoder och attribut.	Inkapsling handlar delvis om att skydda information i en del av programmet från en annan del av programmet.
T4	Med hjälp av delegation blir ett program lättare att förändra.	Delegation gör objekten mer oberoende av varandra (loosely coupled)
T5	Aggregering är att föredra framför komposition om objektet vars beteende man tänker använda har ett egenvärde utanför objektet som använder det.	Vid aggregering instansieras det aggregerade objektet oftast i ägarens konstruktor
T6	Ett UML-interaktionsdiagram ger en dynamisk vy av hur objekten samarbetar. Ett klassdiagram ger istället en statisk vy över samarbetet.	Ett klassdiagram beskriver egenskaper och metoder för klasserna.

Uppgift 1.3

I ett spelprogram förekommer följande deklarationer:

```
class Square { ... } // en ruta pa spelplanen
class GameBoard { // spelplanen
    private Square[] squares; // rutorna
}
class Die { ... } // tarningen
class Player { // en spelare
    private String name;
    private int position;
    private GameBoard board;
    private Die die;
    private Player opponent; // motspelaren
}
class Expert extends Player { ... }
class Novice extends Player { ... }
class Game {
    private Player p1,p2;
    private GameBoard board;
}
```

Beskriv spelet i ett klassdiagram. Visa riktningar för associationerna.

Uppgift 1.4

I följande program brister inkapslingen av data. Åtgärda detta. Lösningen redovisas med Java-kod.

```
public class Point {
  private int x,y;
  public int getX() {
    return x;
  }
  public void setX(int x) {
    this.x = x;
  }
  public int getY() {
    return y;
  }
  public void setY(int y) {
    this.y = y;
  }
}
```

```
public class Circle {
  private Point centre;
  private int radius;
  public Point getCentre() {
    return centre;
  }
  public int getRadius() {
    return radius;
  }
}

public class Drawing {
   private List<Circle> circles;
   //omissions
   private void moveCircle(Circle aC, int dx, int dy) {
     int x1 = aC.getCentre.getX();
     aC.getCentre.setX(x1 + dx);
     int y1 = aC.getCentre.getY();
     aC.getCentre.setY(y1 + dy);
   }
}
```

Uppgift 1.5

Sällskapsspelet Mastermind beskrivs av en tillverkare på följande sätt.

> MASTERMIND is a game which gives each player a chance to outsmart his opponent. The Codemaker secretly sets up a line of Code Pegs behind his shield and the Codebreaker has up to ten opportunities to try and duplicate the colour and each exact position of the hidden Code Pegs without ever seeing them.
>
> The Codemaker secretly puts 4 Code Pegs in 4 holes behind the shield. Use any combination of the six colours. You may use 2 or more Code Pegs of the same colour if you wish.
>
> The Codebreaker will try to duplicate the exact colours and positions of the code hidden behind the shield. Each time the Codebreaker places a row of Code Pegs (they are then left in position troughout the game), the Codemaker must give him the following information by placing the black and white Key Pegs in the Key Peg holes alongside the Code Pegs placed by the Codebreaker, or leaving holes vacant.
>
> **Black Key Pegs** are placed by the Codemaker in any of the Key Peg holes for every Code Peg placed by the Codebreaker which is in the same colour and in exactly the same position as one of the Code Pegs behind the shield.
>
> **White Key Pegs** are placed by the Codemaker in any one of the Key Peg holes when any hidden Code Peg behind the shield matches the Codebreaker's Code Pegs in colour only, but not in position.
>
> **Example:** If one red Code Peg is behind the shield and the Codebreaker places 2 red Code Pegs in the wrong position ONE white Key Peg is used.
>
> If the Codebreaker duplicates the hidden code behind the Codemaker's shield, the Codemaker places 4 black Key Pegs and reveals the hidden code. The games is over.
>
> *Invicta Plastic Ltd, Leicester, England.*

Bestäm lämpliga klasser och associationer för ett program som spelar Mastermind med en person vid datorn som får agera "Codebreaker". Personen styr vad programmet skall göra i nästa steg med kommandon från tangentbordet. Det skall finnas kommandon för att starta ett nytt parti, göra en gissning, inspektera alla tidigare gissningar och resultat, att inspektera den hemliga koden och att avsluta hela spelsessionen.

Kapitel 2

Design smells

Uppgift 2.1
Denna uppgift innehåller ett antal deluppgifter, där varje deluppgift har ett påstående och en anledning. För varje deluppgift, svara med ett av följande alternativ:

A Både påståendet och anledningen är korrekta uttalanden och anledningingen förklarar påståendet på ett korrekt sätt.

B Både påståendet och anledningen är korrekta uttalanden, men anledningen förklarar inte påståendet.

C Påståendet är ett korrekt uttalande, men anledningen är ett felaktigt uttalande.

D Påståendet är felaktigt, men anledningen är ett korrekt uttalande.

E Både påståendet och anledningen är felaktiga uttalanden.

	Påstående	Anledning
T1	Det är svårt att göra ändringar i en stel (rigid) design.	En stel design innebär att det finns många beroenden till en klass.
T2	Upprepning av kod med små variationer skapar onödigt svårunderhållen kod.	Designen kan bli onödigt komplex om man designar för potentiella förändringar och tillägg.
T3	Att designen är ömtålig innebär att delar som skulle kunna återanvändas är för hårt knutna till systemet så att det kostar för mycket att bryta ut dem.	Seghet kan drabba både mjukvaran och miljön och innebär att kostnaden för att bevara designen är högre än kostnaden att bryta mot den och göra ett hack.

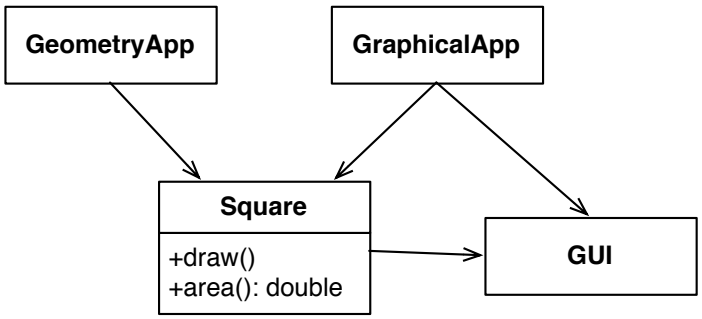
Föreslå en egen teorifråga baserat på Martins presentation av design smells. Formulera din fråga som ett påstående-anledning problem på ovanstående form.

Kapitel 3

Grundläggande designprinciper (SRP, OCP, DIP, DRY)

Uppgift 3.1

a) Square i UML-diagrammet, figur 3.1, har ansvar att rita ut sig själv och att beräkna sin area. Detta bryter mot SRP (eng: Single Responsibility Principle). Ändra designen så att den inte bryter mot SRP. Visa lösningen med ett nytt UML-klassdiagram.



Figur 3.1: Exempel på brott mot SRP

b) Kan brott mot SRP ge problem vid underhåll av kod? Motivera ditt svar kortfattat med hjälp av exemplet ovan och din kunskap om illaluktande design (eng: design smells)

Uppgift 3.2

Figuren nedan till vänster visar en design som bryter mot en av designprinciperna (utöver Open-Closed). Detta upptäcktes när man ville ändra hur utskriften ska göras och det hela rättades till i den högra bilden. Notera att det inte är korrekt Javakod, utan en pseudokod för att visa principen.

```
class Book {

    function getTitle() {
        return "A Great Book";
    }

    function getAuthor() {
        return "John Doe";
    }

    function turnPage() {
        // pointer to next page
    }

    function printCurrentPage() {
        echo "current page content";
    }
}
```

```
class Book {

    function getTitle() {
        return "A Great Book";
    }

    function getAuthor() {
        return "John Doe";
    }

    function turnPage() {
        // pointer to next page
    }

    function getCurrentPage() {
        return "current page content";
    }

}

interface Printer {

    function printPage($page);
}

class PlainTextPrinter implements Printer {

    function printPage($page) {
        echo $page;
    }

}

class HtmlPrinter implements Printer {

    function printPage($page) {
        echo '<div style="single-page">' . $page . '</div>';
    }

}
```

1. Vad heter principen som bryts"i den vänstra designen?
2. Förklara på vilket sätt den vänstra designen bryter mot principen och hur den högra löser problemet.
3. Förklara nackdelarna med den vänstra designen, m.a.p brott mot principen.

Uppgift 3.3

Utforma klasser för att representera punkter och sträckor i planet och i rummet. Punkterna i planet ska representeras med två koordinater och punkter i rummet med tre koordinater. En koordinat ges av en `int`. En sträcka representeras av sina ändpunkter. Det skall finnas en metod för att beräkna längden av en sträcka. Implementeringen skall följa `open/closed-principen` så att man kan lägga till andra sorters punkter, t ex med fler koordinater eller flyttalskoordinater, utan

att ändra i skriven kod. Lösningen presenteras med klassdiagram, där attribut, metoder, synlighet, relationer, multiplicitet och riktning ska finnas med.

Uppgift 3.4

Gör en design av ett paket för att hantera belopp i olika valutor. Uppdragsgivaren kräver att det skall vara möjligt att

- addera, subtrahera och jämföra belopp i en och samma valuta med exakt aritmetik utan att använda andra aritmetiska operationer än addition och subtraktion och utan att behöva skapa nya objekt för varje operation.
- givet ett belopp i en valuta kunna skapa ett belopp i en annan valuta med samma värde. För att kunna göra detta skall varje valuta ha en omräkningsfaktor av typ `double`.
- för varje valuta konstruera belopp baserat på de myntslag som finns, dvs kr och ören för svensk valuta, pund, shilling och pence för en gammal brittisk valuta och lire för den gamla italienska valutan.
- konvertera ett belopp till en sträng på decimalform.

Addition, subtraktion och jämförelser av två belopp i olika valutor skall ge kompileringsfel eller kasta undantag. Lösningen presenteras med ett klassdiagram med ovannämnda valutor och alla generaliseringar, realiseringar, metoder och attribut samt en full Java-implementering av den svenska valutan.

Uppgift 3.5

En tidigare upplaga av en lärobok i objektorienterad programmering och Java innehåller följande kod:

```
...
private Application application;
private JButton b1, b2, b3;
class ActionHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            application.action1();
        } else if (e.getSource() == b2) {
            application.action2();
        } else {
            application.action3();
        }
    }
}
```

Designen strider mot *Open-Closed Principle*; man kan inte lägga till ytterligare `JButton`s utan att modifiera klassen. Gör om designen så att detta blir möjligt. Designen skall innehålla de tre knapparna och händelsehanteringen. Undvik duplicerad kod. Redovisa designen med ett klassdiagram.