OMD – Övning vecka 2

Du förväntas göra lösningsförslag till uppgifterna före övningen, gärna tillsammans med andra kursdeltagare, och får bonus för detta om du är beredd att presentera din lösning inför klassen. Alla problem utom nummer 4 bygger bara på de två första föreläsningarna och avsnitt 2 i UML-häftet.

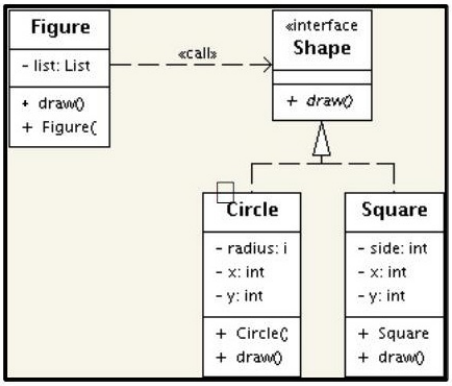
Lösningar till dessa uppgifter presenteras under övningen i läsvecka 2.

---

1 Rita ett klassdiagram som visar alla klasser, gränssnitt, associationer, metoder, konstruerare och attribut som förekommer i följande klass. Gränssnittet `ActionListener` specificeras i `java.awt.event`.

```
public class StartButton extends Button implements ActionListener {
    private Train train;
    public StartButton(Train train) {
        super("Start");
        this.train = train;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent event) {
        train.start();
    }
}
```

2 På torsdagens föreläsning visade läraren



En uppmärksam student kände direkt att designen luktade illa. Klasserna `Circle` och `Square` har två identiska attribut `x` och `y`. Gör om designen så att attributen placeras i en gemensam superklass, lägg till metoden `void move(int dx, int dy)` i gränssnittet och implementera metoden på rätt plats. Metoden `Figure.draw` får ej modifieras. Lösningen redovisas med ett klassdiagram och implementering.

3 Det var ingen som, i föregående uppgift, upptäckte att klassen `Figure` också har en metod med samma namn som en metod i `Shape` och analog semantik. Gör om designen så att `Figure` också implementerar den nya versionen av `Shape`. Implementera `move()` och `draw(Plotter)` där ritning görs på ett objekt av typen `Plotter`:

```
public interface Plotter {
    public void move(int dx, int dy);
    public void penDown();
    public void penUp();
}
```

4 En klass för att representera en summa visas nedan. Klassen för en produkt, `Mul`, är identisk så när som på klassnamnet och operatortecknet.

```
public class Add implements Expr {
    private Expr expr1, expr2;
    public Add(Expr expr1, Expr expr2) {
        this.expr1 = expr1;
        this.expr2 = expr2;
    }
    public String toString() {
        return expr1.toString() + "+" +  expr2.toString();
    }
}
```

Använd *Template method*-mönstret för att eliminera duplicerad kod och duplicerade attribut. Redovisa lösningen med ett klassdiagram och Java-kod.

5 Om man skall konstruera en editor som kan manipulera program är det lämpligt att göra en objektorienterad modell av program. Gör en UML-modell för ett programspråk enligt följande beskrivning. Alla associationer, generaliseringar och attribut skall redovisas, men inga konstruerare eller metoder.

- Ett program är ett block som består av en följd av satser.
- En sats är antingen en tilldelningssats, en if-sats, en skrivsats eller ett block.
- Tilldelningssatsen har en variabel, som är en sträng och ett högerled som är ett uttryck.
- Skrivsatsen innehåller det aritmetiska uttryck vars värde skall skrivas ut.
- En if-sats består av tre delar: ett logiskt uttryck, en then-gren och en else-gren som båda innehller en sats. Else-grenen får utelämnas.
- Ett uttryck är antingen logiskt eller aritmetiskt. Du skall ej närmare redovisa de klasser som används för att representera logiska och aritmetiska uttryck.

6 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.