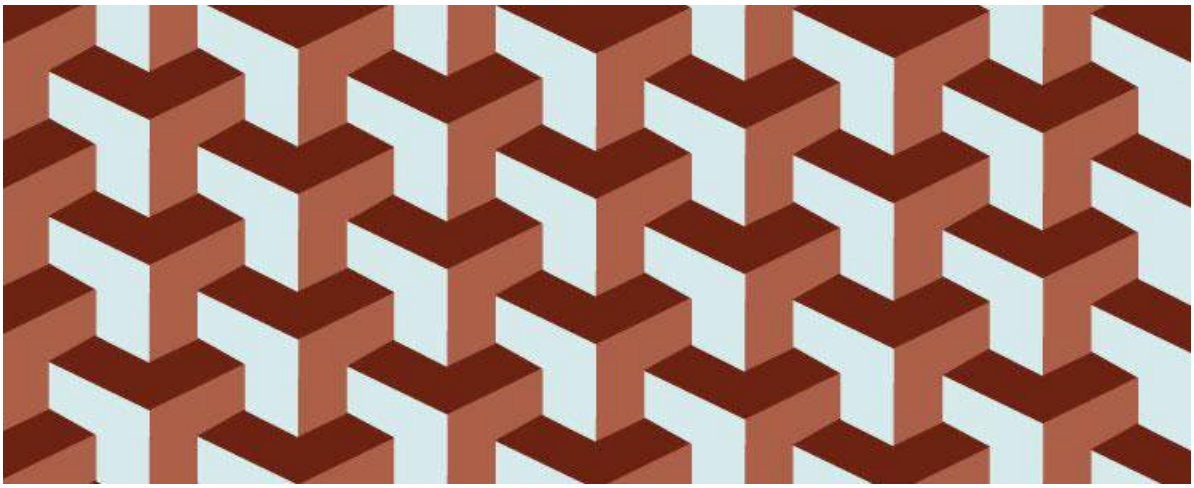
VDAB

samen sterk voor werk

Java
Design Patterns
Takenbundel



Inhoudsopgave

1 TAKEN 2

1.1 Singleton 2

1.2 Singleton 2 2

1.3 Simple factory 2

1.4 Builder 2

1.5 Façade 2

1.6 Adapter 3

1.7 Composite 3

1.8 Decorator 3

1.9 Observer 4

1.10 Strategy 4

2 VOORBEELDOPLOSSINGEN 5

2.1 Singleton 5

2.2 Singleton 2 5

2.3 Simple factory 6

2.4 Builder 7

2.5 Facade 7

2.6 Adapter 9

2.7 Composite 9

2.8 Decorator 11

2.9 Observer 12

2.10 Strategy 13

3 COLOFON 14

1 TAKEN

1.1 Singleton

In de software van een auto is er ook maar één versnellingsbak. Deze kan hoger geschakeld worden (tot vijfde versnelling), lager geschakeld worden (tot neutraal) en achteruit geschakeld worden.

1.2 Singleton 2

Een Magic8Ball geeft op gelijk welke vraag één van volgende random antwoorden:

- Zoals ik het zie, ja.
- Het is zeker.
- Hoogst waarschijnlijk.
- Ziet er goed uit.
- Zonder twijfel.
- Ja.
- Zeker.
- Je mag er op rekenen.
- Vraag dit later nog eens.
- Dit wil je niet weten.
- Ik kan dit nu niet voorspellen.
- Concentreer je en stel je vraag opnieuw.
- Je moet er niet op rekenen.
- Nee.
- Ziet er niet goed uit.
- Zeer twijfelachtig.

Verzeker

- dat slechts één Magic8Ball instance in je applicatie voorkomt.
- dat deze geen twee keer na mekaar hetzelfde antwoord geeft.

1.3 Simple factory

De Kerstman beslist welk cadeau 's hij koopt op basis van de leeftijd van de kinderen:

- Kinderen tot en met 5 jaar krijgen een pop.
- Kinderen van 6 tot 12 jaar krijgen een gezelschapsspel.
- Kinderen ouder dan 12 jaar krijgen een boekenbon.

Elk stuk speelgoed heeft een prijs.

1.4 Builder

Een coordinaat uit de wiskunde heeft drie parameters: x, y en z.
Maak een builder, zodat een coordinaat stapsgewijs en leesbaar kan gemaakt worden.

1.5 Façade

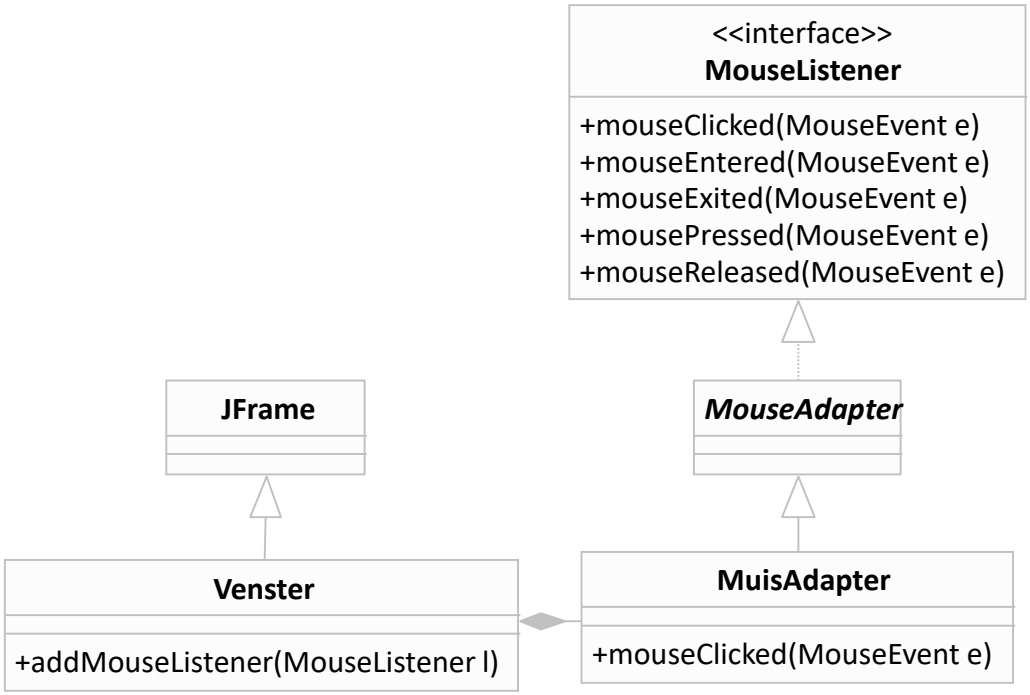
Als een stereoketen gestart wordt, moet de versterker, de equalizer en het laatst gestopte apparaat starten. Dit kan de radio zijn of de CD speler. De stereoketen kan daarna de radio starten. De stereoketen kan ook de CD speler starten. Als je één van beide apparaten start, stop je het andere apparaat. Als je de stereoketen stopt, stop je de versterker, de equalizer en de radio of CD speler, afhankelijk welke van beide laatst gestart werd.

Zorg er voor dat de class Main niet veel intelligentie bevat, wel de class Stereoketen (die dient als façade).

Als een apparaat gestart of gestopt wordt, toon je een zin op de console (bvb. Versterker gestart)

1.6 Adapter

Je maakt met Swing een venster. Als de gebruiker klikt in dit venster, toon je in een popup venster de X coordinaat van de muisaanwijzer. Normaal moet je hiertoe op je venster class de method `addMouseListener()` uitvoeren en een object meegeven waarvan de class de interface `MouseListener` implementeert. Die interface bevat de method `mouseClicked` die je nodig hebt in de oefening. Daarnaast bevat de interface nog vier andere methods die je niet nodig hebt.
Als je de interface zelf implementeert, moet je echter alle methods uit de interface implementeren, ook degene die je niet nodig hebt. Als oplossing bevat Java de class `MouseAdapter`.
Deze adapter class implementeert ook de interface `MouseListener`. De implementatie van elke method bestaat uit een lege method. Om te reageren op het `mouseClicked` event volstaat het een class te maken die erft van `MouseAdapter`, het `MouseClicked` event te overriden
en een object van deze class mee te geven aan `addMouseListener`.



Voer deze stappen uit.

1.7 Composite

Een project van een bouwfirma bevat grondstofkosten, arbeidskosten en subprojecten.
Een subproject bevat op zijn beurt grondstofkosten, arbeidskosten en eventuele subprojecten.

Een grondstofkost bestaat uit de eenheidsprijs van de grondstof en de hoeveelheid grondstof.
Een arbeidskost bestaat uit het uurloon en het aantal gepresteerde uren.

Je maakt een applicatie waarmee je een project kan opbouwen en de totale prijs van dit project kan berekenen.

1.8 Decorator

Een arrangement van een vakantiepark bestaat uit een bungalow. Optioneel kan daar een barbecue aan toegevoegd worden. Optioneel kunnen ook fietsen toegevoegd worden.
Ieder onderdeel heeft een taak voor het personeel van het vakantiepark:

Onderdeel	**Taak**
bungalow	kuis de bungalow
barbecue	plaats een barbecue bij de bungalow
fietsen	plaats fietsen bij de bungalow

Je vraagt in de Main of de gebruiker een barbecue en/of fietsen wil in zijn arrangement.
Je toont daarna de taken die het personeel voor dat arrangement moeten uitvoeren.

1.9 Observer

Een fotokopiemachine heeft een serienummer en kan kopies maken.
Om de x kopies heeft een fotokopiemachine onderhoud nodig.
Techniekers worden hiervan op de hoogte gebracht. Elke technieker heeft een personeelsnummer.

1.10 Strategy

Afhankelijk van de temperatuur en de wind, neem je een ander voertuig om een meer over te steken: ijsschaatsen, zeilen, motorboot

Je moet bij iedere oversteek van het meer drie handelingen doen:

1. Inpakken
2. Het vervoer zelf
3. Uitpakken

Mogelijke output:

```
Inpakken
Schaatsen
Uitpakken
```

2 VOORBEELDOPLOSSINGEN

2.1 Singleton

```
package be.vdab;
public enum Versnellingsbak {
  INSTANCE;
  private int versnelling;
  public void hoger() {
    if (versnelling == 5) {
      throw new VerkeerdeVersnellingException();
    }
    versnelling++;
  }
  public void lager() {
    if (versnelling == -1) {// achteruit
      throw new VerkeerdeVersnellingException();
      }
    versnelling--;
  }
}
```

```
package be.vdab;
public class VerkeerdeVersnellingException extends RuntimeException {
  private static final long serialVersionUID = 1L;
}
```

```
package be.vdab;
public class Main {
  public static void main(String[] args) {
    try {
      Versnellingsbak.INSTANCE.hoger();
      Versnellingsbak.INSTANCE.lager();
    } catch (VerkeerdeVersnellingException ex) {
      System.out.println("verkeerde versnelling");
    }
  }
}
```

2.2 Singleton 2

```
package be.vdab;
import java.util.Random;
public enum Magic8Ball {
  INSTANCE;
  private final String[] antwoorden = new String[] { "Zoals ik het zie, ja.",
    "Het is zeker.", "Hoogst waarschijnlijk.",      "Ziet er goed uit.",
    "Zonder twijfel.", "Ja.", "Zeker.", "Je mag er op rekenen.",
    "Vraag dit later nog eens.", "Dit wil je niet weten.",
    "Ik kan dit nu niet voorspellen.",
    "Concentreer je en stel je vraag opnieuw.", "Je moet er niet op rekenen.",
    "Nee.", "Ziet er niet goed uit.", "Zeer twijfelachtig." };
  private int vorigAntwoordIndex = -1;
  public String getAntwoord(String vraag) {
    System.out.println("Vraag:" + vraag);
    Random random = new Random();
    int antwoordIndex = random.nextInt(antwoorden.length);
    while (antwoordIndex == vorigAntwoordIndex) {
      antwoordIndex = random.nextInt(antwoorden.length);
    }
    vorigAntwoordIndex = antwoordIndex;
```

```
    return "Antwoord:"+antwoorden[antwoordIndex];
  }
}
```

```
package be.vdab;
public class Main {
  public static void main(String[] args) {
    System.out.println(
      Magic8Ball.INSTANCE.getAntwoord("Win ik morgen de lotto ?"));
    System.out.println(
      Magic8Ball.INSTANCE.getAntwoord("Zal het morgen regenen ?"));
  }
}
```

2.3 Simple factory

```
package be.vdab;
import java.math.BigDecimal;
public enum SpeelgoedFactory {
  INSTANCE;
  public Speelgoed kiesSpeelgoed(int leeftijd) {
    if (leeftijd < 6) {
      return new Pop(BigDecimal.valueOf(15));
    } else if (leeftijd < 13) {
      return new Gezelschapsspel(BigDecimal.valueOf(25));
    } else {
      return new Boekenbon(BigDecimal.valueOf(27));
    }
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public abstract class Speelgoed {
  private BigDecimal prijs;
  public Speelgoed(BigDecimal prijs) {
    this.prijs = prijs;
  }
  public BigDecimal getPrijs() {
    return prijs;
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public class Pop extends Speelgoed {
  public Pop(BigDecimal prijs) {
    super(prijs);
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public class Gezelschapsspel extends Speelgoed {
  public Gezelschapsspel(BigDecimal prijs) {
    super(prijs);
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public class Boekenbon extends Speelgoed {
  public Boekenbon(BigDecimal prijs) {
    super(prijs);
  }
}
```

```
package be.vdab;
public class Main {
  public static void main(String[] args) {
    Speelgoed speelgoed = SpeelgoedFactory.INSTANCE.kiesSpeelgoed(15);
    System.out.println(speelgoed.getPrijs());
  }
}
```

2.4 Builder

```
package be.vdab;
public class Coordinaat {
  private final int x, y, z;
  private Coordinaat(int x, int y, int z) {
    this.x = x;
    this.y = y;
    this.z = z;
  }
  @Override
  public String toString() {
    return x + " " + y + " " + z;
  }
  public static class CoordinaatBuilder {
    private int x, y, z;
    public CoordinaatBuilder metX(int x) {
      this.x = x;
      return this;
    }
    public CoordinaatBuilder mety(int y) {
      this.y = y;
      return this;
    }
    public CoordinaatBuilder metZ(int z) {
      this.z = z;
      return this;
    }
    public Coordinaat maakCoordinaat() {
      return new Coordinaat(x, y, z);
    }
  }
}
```

2.5 Facade

```
package be.vdab;
public class Versterker {
  public void start() {
    System.out.println("versterker gestart");
  }
  public void stop() {
    System.out.println("versterker gestopt");
  }
}
```

```java
package be.vdab;
public class Equalizer {
  public void start() {
    System.out.println("equalizer gestart");
  }
  public void stop() {
    System.out.println("equalizer gestopt");
  }
}
```

```java
package be.vdab;
public interface Geluidsbron {
  void start();
  void stop();
}
```

```java
package be.vdab;
public class Radio implements Geluidsbron {
  @Override
  public void start() {
    System.out.println("radio gestart");
  }
  @Override
  public void stop() {
    System.out.println("radio gestopt");
  }
}
```

```java
package be.vdab;
public class CDSpeler implements Geluidsbron{
  @Override
  public void start() {
    System.out.println("CD speler gestart");
  }
  @Override
  public void stop() {
    System.out.println("CD speler gestopt");
  }
}
```

```java
package be.vdab;
public class Stereoketen {
  private Versterker versterker = new Versterker();
  private Equalizer equalizer = new Equalizer();
  private Radio radio = new Radio();
  private CDSpeler cdSpeler = new CDSpeler();
  private Geluidsbron huidigeGeluidsbron;
  private boolean gestart;
  public void start() {
    versterker.start();
    equalizer.start();
    if (huidigeGeluidsbron != null) {
      huidigeGeluidsbron.start();
    }
    gestart = true;
  }
  public void stop() {
    versterker.stop();
    equalizer.stop();
    if (huidigeGeluidsbron != null) {
      huidigeGeluidsbron.stop();
```

```
    }
    gestart = false;
  }
  public void startRadio() {
    if (gestart) {
      if (huidigeGeluidsbron != null) {
        huidigeGeluidsbron.stop();
      }
      radio.start();
      huidigeGeluidsbron = radio;
    }
  }
  public void startCDSpeler() {
    if (gestart) {
      if (huidigeGeluidsbron != null) {
      huidigeGeluidsbron.stop();
      }
      cdSpeler.start();
      huidigeGeluidsbron = cdSpeler;
    }
  }
}
```

2.6 Adapter

```
package be.vdab;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class Venster extends JFrame {
  private static final long serialVersionUID = 1L;
  public Venster() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setExtendedState(JFrame.MAXIMIZED_BOTH);
    addMouseListener(new MuisAdapter());
  }
  private class MuisAdapter extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent event) {
      JOptionPane.showMessageDialog(Venster.this, String.valueOf(event.getX()));
    }
  }
}
```

```
package be.vdab;
public class Main {
  public static void main(String[] args) {
    new Venster().setVisible(true);
  }
}
```

2.7 Composite

```
package be.vdab;
import java.math.BigDecimal;
public interface Kost {
  BigDecimal getBedrag();
}
```

```
package be.vdab;
import java.math.BigDecimal;
```

```
public class GrondstofKost implements Kost {
  private BigDecimal eenheidsprijs;
  private BigDecimal hoeveelheid;
  public GrondstofKost(BigDecimal eenheidsprijs, BigDecimal hoeveelheid) {
    this.eenheidsprijs = eenheidsprijs;
    this.hoeveelheid = hoeveelheid;
  }
  @Override
  public BigDecimal getBedrag() {
    return eenheidsprijs.multiply(hoeveelheid);
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public class Arbeidskost implements Kost {
  private BigDecimal uurloon;
  private BigDecimal aantalUren;
  public Arbeidskost(BigDecimal uurloon, BigDecimal aantalUren) {
    this.uurloon = uurloon;
    this.aantalUren = aantalUren;
  }
  @Override
  public BigDecimal getBedrag() {
    return uurloon.multiply(aantalUren);
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
public class Project implements Kost {
  private final List<Kost> kosten = new ArrayList<>();
  public void addKost(Kost kost) {
    kosten.add(kost);
  }
  @Override
  public BigDecimal getBedrag() {
    BigDecimal bedrag = BigDecimal.ZERO;
    for (Kost kost : kosten) {
      bedrag = bedrag.add(kost.getBedrag());
    }
    return bedrag;
  }
}
```

```
package be.vdab;
import java.math.BigDecimal;
public class Main {
  public static void main(String[] args) {
    Project project = new Project();
    project.addKost(new GrondstofKost(
     BigDecimal.valueOf(1000), BigDecimal.valueOf(3)));
    project.addKost(new Arbeidskost(
      BigDecimal.TEN, BigDecimal.valueOf(100)));
    Project subProject = new Project();
    subProject.addKost(new GrondstofKost(
      BigDecimal.valueOf(1000), BigDecimal.valueOf(4)));
    subProject.addKost(new Arbeidskost(
      BigDecimal.TEN, BigDecimal.valueOf(200)));
```

```
    project.addKost(subProject);
    System.out.println(project.getBedrag());
  }
}
```

2.8 Decorator

```
package be.vdab;
public interface Arrangement {
  String getTaken();
}
```

```
package be.vdab;
public class Bungalow implements Arrangement {
  @Override
  public String getTaken() {
    return "kuis de bungalow";
  }
}
```

```
package be.vdab;
public abstract class ArrangementDecorator implements Arrangement {
  protected final Arrangement gedecoreerdArrangement;
  public ArrangementDecorator(Arrangement gedecoreerdArrangement) {
    this.gedecoreerdArrangement = gedecoreerdArrangement;
  }
}
```

```
package be.vdab;
public class MetBarbeque extends ArrangementDecorator {
  public MetBarbeque(Arrangement gedecoreerdArrangement) {
    super(gedecoreerdArrangement);
  }
  @Override
  public String getTaken() {
    return super.gedecoreerdArrangement.getTaken()
      + ", plaats een BBQ bij de bungalow";
  }
}
```

```
package be.vdab;
public class MetFietsen extends ArrangementDecorator {
  public MetFietsen(Arrangement gedecoreerdArrangement) {
    super(gedecoreerdArrangement);
  }
  @Override
  public String getTaken() {
    return super.gedecoreerdArrangement.getTaken()
      + ", plaats fietsen bij de bungalow";
  }
}
```

```
package be.vdab;
import java.util.Scanner;
public class Main {
  public static void main(String[] args) {
    Arrangement arrangement = new Bungalow();
    try (Scanner scanner = new Scanner(System.in)) {
      System.out.print("Barbeque (j/n):");
      if (scanner.next().equals("j")) {
        arrangement = new MetBarbeque(arrangement);
      }
      System.out.print("Fietsen (j/n):");
```

```
      if (scanner.next().equals("j")) {
        arrangement = new MetFietsen(arrangement);
      }
      System.out.println(arrangement.getTaken());
    }
  }
}
```

2.9 Observer

```
package be.vdab;
import java.util.Observable;
public class Fotokopiemachine extends Observable {
  private final static int ONDERHOUD_OM_DE_PAGINAS = 10_000;
  private final long serienummer;
  private int aantalPaginasSindsLaatsteOnderhoudsbeurt;
  public Fotokopiemachine(long serienummer) {
    this.serienummer = serienummer;
  }
  public void kopieer(int paginas) {
    System.out.println(paginas + " fotokopie(s)");
    aantalPaginasSindsLaatsteOnderhoudsbeurt += paginas;
    if (aantalPaginasSindsLaatsteOnderhoudsbeurt >= ONDERHOUD_OM_DE_PAGINAS) {
      setChanged();
      notifyObservers();
    }
  }
  public long getSerieNummer() {
    return serienummer;
  }
  public void doeOnderhoud() {
    aantalPaginasSindsLaatsteOnderhoudsbeurt = 0;
  }
}
```

```
package be.vdab;
import java.util.Observable;
import java.util.Observer;
public class Technieker implements Observer {
  private final long personeelsnummer;
  public Technieker(long personeelsnummer) {
    this.personeelsnummer = personeelsnummer;
  }
  @Override
  public void update(Observable observable, Object object) {
    if (! (observable instanceof Fotokopiemachine)) {
      throw new IllegalArgumentException();
    }
    Fotokopiemachine machine = (Fotokopiemachine) observable;
    System.out.println("Technieker " + personeelsnummer +
      " noteert onderhoud voor machine " + machine.getSerieNummer());
    /* De technieker doet later het onderhoud van de machine, want
       op dit moment kunnen ook andere techniekers op de hoogte gebracht
       worden van deze machine en het is niet de bedoeling dat meerdere
       techniekers tegelijk die machine onderhouden
       (zie voorbeeld in Main met één machine en twee techniekers */
  }
}
```

```
package be.vdab;
public class Main {
  public static void main(String[] args) {
    Fotokopiemachine machine = new Fotokopiemachine(123);
    Technieker technieker1 = new Technieker(1);
    Technieker technieker2 = new Technieker(2);
    machine.addObserver(technieker1);
    machine.addObserver(technieker2);
    machine.kopieer(5000);
    machine.kopieer(5000);
  }
}
```

2.10 Strategy

```
package be.vdab;
@FunctionalInterface
public interface Vervoermiddel {
  void vervoer();
}
```

```
package be.vdab;
public class Vervoer {
  public void doeEenOversteek(Vervoermiddel vervoermiddel) {
    System.out.println("Inpakken");
    vervoermiddel.vervoer();
    System.out.println("Uitpakken");
  }
}
```

```
package be.vdab;
public class Main {
public static void main(String[] args) {
  Vervoer vervoer = new Vervoer();
  vervoer.doeEenOversteek(() -> System.out.println("Schaatsen over het ijs"));
  System.out.println();
  vervoer.doeEenOversteek(() -> System.out.println("Zeilen over het meer"));
  }
}
```

3 COLOFON

Domeinexpertisemanager: Jean Smits

Moduleverantwoordelijke: Hans Desmet

Medewerkers: Hans Desmet

Versie: 20/8/2018

Nummer dotatielijst: