Übung Java Document Class

UML Klassendiagramm

Mit dieser Übung implementieren wir das folgende UML Klassendiagramm in Java:

Klasse Document

Wir erstellen die Java Klasse Document gemäss dem folgenden Diagramm:Erstellen Sie die Java Klasse Document in der Datei Document.java und programmieren Sie das Klassengerüst (z.B. wie folgt):package ch.std.jnoo.document;

public class Document {

}Definieren Sie nun in der Klasse Document die Attribute Titel (title) und Autor (author) vom Type String mit private Modifier (z.B. wie folgt):private String title;
private String author;Die Klasse Document soll nun noch über den Default Konstruktor (ohne Argumente) verfügen. Weiter soll die Klasse einen Konstuktor enthalten, der jedes Objekt mit Titel und Autor initialisieren kann (z.B. wie folgt):public Document()
{
  this ("unknown", "unknown");
}
public Document(String title, String author)
{
  this.title = title;
  this.author = author;
}Java Klassen definieren oft die Methode toString(). Die Methode soll den Zustand der Attribute als String zurückgeben. Die Methode toString() könnte für die Klasse Document z.B. wie folgt definiert werden:public String toString()
{
  return "title: " + title + ", author: " + author;
}Nun erstellen wir eine neue Klasse für das Testen unserer Document Klasse wie folgt:package ch.std.jnoo.document.test;

import ch.std.jnoo.document.Document;

public class DocumentTest {

public static void main(String[] args) {
    Document document = new Document("Java Basics", "Any");
    System.out.println(document.toString());
  }
}Die Klasse DocumentTest lässt sich nun über die Methode main(...) ausführen. Das ergibt den folgenden Output an die Konsole:Wir haben mit diesem Schritt das folgende Klassendiagramm implementiert:

Klasse Page

Nun erstellen wir die Klasse Page gemäss dem folgenden Diagramm:Erstellen Sie die Klasse Page in der Datei Page.java und programmieren Sie das Klassengerüst (z.B. wie folgt):package ch.std.jnoo.document;

public class Page {

}Definieren Sie nun in der Klasse Page das Attribut Nummer (number) vom Typ int mit private Modifier (z.B. wie folgt):private int number;Ergänzen Sie die Klasse mit den geeigneten Konstruktoren und mit der Methode toString().Die Klassen Document und Page enthalten diverse Attribute, welche private deklariert sind. Diese Attribute sind von aussen her nicht sichtbar. Wir können auf diese Daten nicht zugreifen. Wir benötigen nun noch Methoden, welche den kontrollierten Zugriff auf diese Attribute ermöglichen. Java benennt solche Zugriffsmethoden auf Attribute Getter und Setter Methoden. Das Verfahren hierzu ist einfach: Getter Methoden sind public und beginnen mit der Kennung "get" gefolgt vom Attributnamen, dessen erster Buchstabe gross geschrieben wird. Als return-Wert wird der Typ des Attributes zurückgegeben. Die Getter-Methode zum Attribut number der Klasse Page sieht wie folgt aus:public int getNumber() {
  return number;
}Setter Methoden sind public und beginnen mit der Kennung "set" gefolgt vom Attributnamen, dessen erster Buchstabe gross geschrieben wird. Es wird kein return-Wert zurückgegeben. Die Setter-Methode zum Attribut number der Klasse Page sieht wie folgt aus:public void setNumber(int number) {
  this.number = number;
}Definieren Sie zu den Klassen Document und Page die entsprechenden Setter- und Getter-Methoden.Nun fehlt noch die Komposition oder Assoziation zwischen Document und Page. Es handelt sich hier um eine 1:n Beziehung, welche am besten über eine Liste abgebildet wird. Hierzu bietet Java diverse Möglichkeiten über Collections. Das Klassendiagram der Collection Klassen von Java finden Sie hier:Wir verwenden für unsere Beziehung in der Klasse Document ein Attribut "pageList" vom Type java.util.List. Im Document Konstruktor weisen wir dem Attribut eine Instanz der Klasse java.util.ArrayList zu. Nun sollten wir noch in der Lage sein, Page Instanzen dem Dokument hinzuzufügen und auch wieder zu entfernen. Hierzu definieren wir in der Klasse Document die folgenden Methoden:public void addPage (Page page) {
  pageList.add(page);
}
public void removePage(Page page) {
  pageList.remove(page);
}Wir passen nun die Klasse DocumentTest an die neue Situation an und ändern diese wie folgt:package ch.std.jnoo.document.test;

import ch.std.jnoo.document.Document;
import ch.std.jnoo.document.Page;
public class DocumentTest {

public static void main(String[] args) {
    Document document = new Document("Java Basics", "Any");
    document.addPage(new

Page(1));
  document.addPage(new Page(2));
  document.addPage(new Page(3));
  document.print();
  }
}Sicher ist Ihnen die Methode print() aufgefallen. Diese Methode soll das gesamte Dokument an der Systemkonsole (System.out) ausdrucken. Das folgende Listing zeigt die print() Methode der Klasse Document:public void print() {
  System.out.println("document print");
  System.out.println("title: " + title);
  System.out.println("author: " + author);
  for (Page page : this.pageList) {
  page.print();
  }
}Ergänzen Sie auch die Klasse Page mit einer geeigneten print() Methode, welche einfach die Seitennummer ausgibt. Der nachfolgende ScreenShot zeigt die Ausgabe der Testanwendung:Wir haben nun das folgende Klassendiagramm implementiert:

Klasse Element

Nun erstellen wir die Klasse Element gemäss dem folgenden Diagramm:Da wir noch nicht genau wissen, was ein Element konkret ist, definieren wir die Klasse abstrakt (abstract) in der Datei Element.java und programmieren Sie das Klassengerüst (z.B. wie folgt):package ch.std.jnoo.document;

public abstract class Element {

}Abstrakte Klassen in Java sind als solche zu markieren und enthalten in der Regel mindestens eine abstrakte Methode. Abstrakte Methoden werden nur angedeutet und nicht implementiert. Wir benötigen für die Klasse Element zur Zeit die abstrakte Methode print() wie folgt:public abstract void print();Eine abstrakte Klasse ist in Java nicht instanziierbar. Solche Klassen werden durch Subklassen abgeleitet, welche alsdann in der Regel die abstrakten Methoden der Superklasse implementieren.Nun fehlt noch die Komposition oder Assoziation zwischen Page und Element. Es handelt sich hier um eine 1:n Beziehung, welche wiederum am besten über eine Liste abgebildet wird.Wir verwenden für unsere Beziehung in der Klasse Page ein Attribut "elementList" vom Type java.util.List. Im Page Konstruktor weisen wir dem Attribut eine Instanz der Klasse java.util.ArrayList zu. Programmieren Sie in der Klasse auch noch die entsprechenden Methoden für die Verwaltung der Elemente (addElement/removeElement).Für den Test dieses Schrittes können wir auf die Klasse DocumentTest des letzten Schrittes Page zurückgreifen und diese Klasse unverändert wiederverwenden. Das Testresultat sollte gleich bleiben.Wir haben nun somit das folgende Klassendiagramm implementiert:

Shape Klassen

Nun erstellen wir die Shape Klassen gemäss dem folgenden Diagramm: Alle Shape Klassen sind aus der Klasse Element abgeleitet und implementieren die Methode print(). Weiter enthalten die Klassen Attribute gemäss Diagramm und die entsprechenden Setter und Getter Methoden. Alle Shape Klassen liegen im gleichen Package. Jede Klasse implementiert die Methode toString() und definiert die sinnvollen Konstruktoren.Als Beispiel zeigt das folgende Listing die Klasse Line:package ch.std.jnoo.document;

public class Line extends Element {
  private int x1;
  private int y1;
  private int x2;
  private int y2;
  public Line() {
  this(0, 0, 0, 0);
  }
  public Line(int x1, int y1, int x2, int y2) {
  this.x1 = x1;
  this.y1 = y1;
  this.x2 = x2;
  this.y2 = y2;
  }
  public int getX1() {
  return x1;
  }
  public int getX2() {
  return x2;
  }
  public int getY1() {
  return y1;
  }
  public int getY2() {
  return y2;
  }
  public void setX1(int x1) {
  this.x1 = x1;
  }
  public void setX2(int x2) {
  this.x2 = x2;
  }
  public void setY1(int y1) {
  this.y1 = y1;
  }
  public void setY2(int y2) {
  this.y2 = y2;
  }
  public void print() {
  System.out.println("line [x1:" + x1 + ",y1:" + y1 + ",x2:" + x2 + ",y2:" + y2 + "]");
  }
  public String toString() {
  return "x1: " + x1 + ", y1: " + y1 + ", x2: " + x2 + ", y2: " + y2;"
  }
}Implementieren Sie alle Shape Klassen.Für das Testen der neuen Klassen benötigen wir ein anderes Testprogramm. Das folgende Listing zeigt ein mögliches Testprogramm auf:package ch.std.jnoo.document.test;

import ch.std.jnoo.document.Document;
import ch.std.jnoo.document.Line;
import ch.std.jnoo.document.Oval;
import ch.std.jnoo.document.Page;
import ch.std.jnoo.document.Rect;
import ch.std.jnoo.document.Text;

public class DocumentTest {

  public static void main(String[] args) {
  Document document = new Document("Java Basics", "Any");
  Page one = new Page(1);
  document.addPage(one);
  one.addElement(new Text("this is text"));
  one.addElement(new Line(1, 1, 5, 5));
  one.addElement(new Rect(1, 1, 50, 50));
  one.addElement(new Oval(100, 10, 50, 50));
  document.print();
  }
}Das Programm zeigt jedoch die einzelnen

Elemente nicht an (siehe ScreenShot unten):Was ist da falsch? Nehmen Sie die entsprechenden Korrekturen vor, so dass die Shapes über die Methode print() ausgegeben werden (z.B. so):Wir haben nun das folgende Klassendiagramm implementiert:

Klasse Group

Nun fehlt noch die Gruppenbeziehung über die Klasse Group, welche gemäss dem folgenden Diagramm implementiert werden soll:Programmieren Sie nun die Klasse Group, so dass das folgende Testprogramm korrekt funktioniert:package ch.std.jnoo.document.test;

import ch.std.jnoo.document.Document;
import ch.std.jnoo.document.Group;
import ch.std.jnoo.document.Line;
import ch.std.jnoo.document.Oval;
import ch.std.jnoo.document.Page;
import ch.std.jnoo.document.Rect;
import ch.std.jnoo.document.Text;

public class DocumentTest {

  public static void main(String[] args) {
    Document document = new Document("Java Basics", "Any");
    Page one = new Page(1);
    document.addPage(one);
    Group g1 = new Group();
    one.addElement(g1);
    g1.addElement(new Text("this is text for group 1"));
    g1.addElement(new Line(1,1,5,5));
    g1.addElement(new Rect(1,1,50,50));
    g1.addElement(new Oval(100,10,50,50));
    Group g2 = new Group();
    one.addElement(g2);
    g2.addElement(new Text("this is text for group 2"));
    document.print();
  }
}

Lösung

Sie finden die Lösung hier.

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/kurse-java-oo-lernen-kurs-8-für-nicht-ressourcen