13.11 Übungsblatt 11

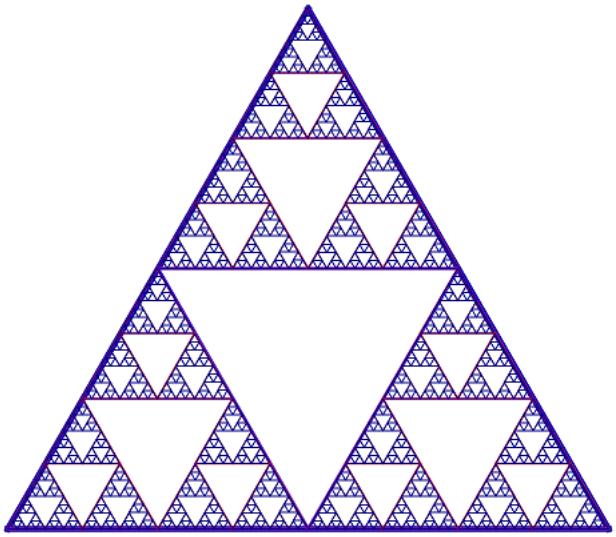


Bild 13.7: Auch Grafiken kann man überschreiben ...

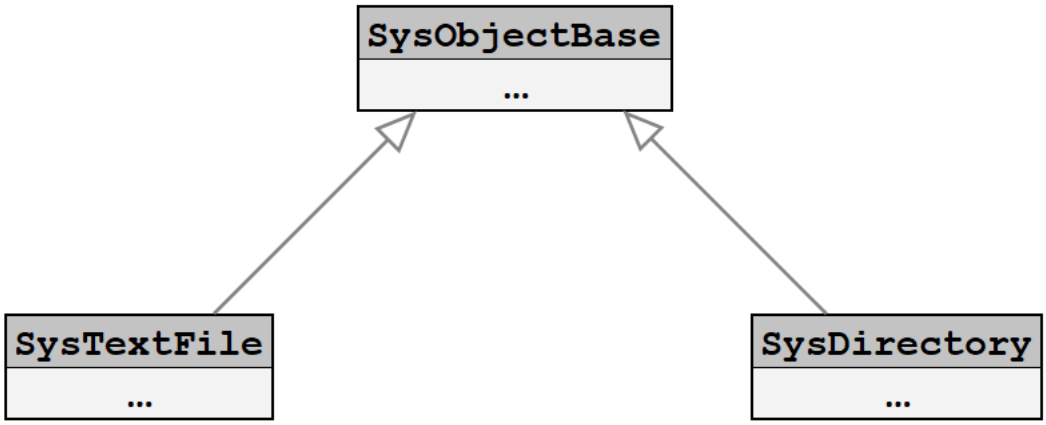
In diesem Blatt soll mit Ableitungen, Klassenhierarchien und dem Information-Hiding gearbeitet werden.

13.11.1 Eine Klassenhierarchie implementieren

Die erste Version von Windows NT brachte viele Neuerungen mit, wie zum Beispiel die Hardware Abstraction Layer *(HAL)* und den *Object Manager*. Dieser Object Manager verwaltet alle Ressourcen des Betriebssystems. Dort gibt es eine Basisklasse mit allgemeinen Angaben, wie zum Beispiel dem Namen der Ressource. Davon wurden Spezialisierungen abgeleitet wie

- Geräte wie Drucker,
- Verzeichnisse,
- Dateien,
- ... .

Diese Vorgehensweise soll hier in dieser Aufgabe in Java nachempfunden werden. Die zu implementierende Klassenhierarchie sieht dabei so aus:



Die Funktion der Klassen ist hierbei:

SysObjectBase:
Basisklasse mit Namen und Besitzer der Ressource.

SysTextFile:
Textdateien mit zusätzlicher Information über den Dateityp und den Inhalt der Datei.

SysDirectory:
Ein Verzeichnis von Ressourcen (also Objekten, die von `SysObjectBase` abgeleitet sind). Da in der Aufgabe nur Textdateien betrachtet werden, also entweder Objekte der Klasse `SysTextFile` oder Objekte der Klasse `SysDirectory`. Die Verzeichnisse sind also *rekursiv*.

Aufgabe 11.1: *Basisklasse für alle Systemobjekte*

Entwerfen und implementieren Sie bitte die Basisklasse für alle Systemobjekte. Beachten Sie dabei folgende Randbedingungen:

1. Die Klasse soll den Namen `SysObjectBase` bekommen.
2. Die Klasse hat zwei `String`-Instanzvariablen: `name` für den Namen der Ressource und `owner` für den Besitzer (als Login-Name, z.B. `brul0001`) der Ressource.
3. Der Konstruktor soll lediglich den Namen der Ressource als Parameter erhalten. Der Besitzer soll im Konstruktor durch `System.getenv("USERNAME")` ermittelt werden.
4. Verhindern Sie bitte nach der Testphase, dass Objekte dieser Klasse instanziiert werden können.
5. Überschreiben Sie bitte die `toString`-Methode, so dass der Klassenname und die Instanzvariablen als `String` zurückgegeben werden, zum Beispiel so: `Class=SysObjectBase, Name=Hello, User=Ulrich`.

Aufgabe 11.2: ***Klasse für Textdateien***

Verfeinern Sie bitte die Basisklasse für alle Systemobjekte `SysObjectBase` so, dass diese Textdateien verwalten kann. Beachten Sie dabei folgende Randbedingungen:

1. Die Klasse soll den Namen `SysTextFile` bekommen.
2. Die Klasse hat zwei weitere `String`-Instanzvariablen:
   - `type`, der Dateityp, zum Beispiel `java` oder `txt` und
   - `text`, der Inhalt der Datei.
3. Der Konstruktor soll den Namen und Typ der Textdatei als Parameter erhalten.
4. Überschreiben Sie bitte wieder die `toString`-Methode, so dass zusätzlich zur Information der Basisklasse noch der Dateityp und die Länge der Datei als `String` zurückgegeben werden, zum Beispiel so:
   `Class=SysTextFile, Name=Hello, User=Ulrich, Type=java, Length=32`.
5. Das folgende `main`

   ```
   public static void main(String[] args) {
     SysTextFile stf = new SysTextFile("Hello", "java");
     stf.setText("public class HelloWorld{/*...*/}");
     System.out.printf("%s:\n  %s\n\n", stf, stf.getText() );
   }
   ```

   sollte folgende Ausgabe ergeben:

   ```
   Class=SysTextFile, Name=Hello, User=Ulrich, Type=java, Length=32:
     public class HelloWorld{/*...*/}
   ```

Aufgabe 11.3: ***Klasse für Verzeichnisse***

Verfeinern Sie bitte die Basisklasse für alle Systemobjekte `SysObjectBase` so, dass diese Verzeichnisse verwalten kann. Beachten Sie dabei folgende Randbedingungen:

1. Die Klasse soll den Namen `SysDirectory` bekommen.
2. Die Klasse hat eine weitere Instanzvariable:
   `SysObjectBase[] sysObjects;`. Es werden also in einem Array alle Objekte vom Typ `SysObjectBase` gehalten, die zu diesem Verzeichnis gehören. Da die Klasse `SysDirectory` von `SysObjectBase` abgeleitet ist, kann solch ein Objekt auch ein weiteres Verzeichnis sein.
3. Der Konstruktor soll den Namen und das Array mit den Objekten so erhalten:
   `SysDirectory(String name, SysObjectBase ... sysObjects)`.
   Es soll also eine Parameterliste variabler Länge eingesetzt werden. Wie das geht, kann man in der Klasse `methods.StudentGroup` aus Kapitel 7 studieren.

4. Überschreiben Sie bitte wieder die `toString`-Methode, so dass zusätzlich zur Information der Basisklasse noch die Anzahl der Objekte im Verzeichnis ausgegeben wird, zum Beispiel so:
   `Class=SysDirectory, Name=HOME, User=brul0001, NumberOfObjects=2`.
5. Das folgende `main`

```
public static void main(String[] args) {
  SysTextFile hello = new SysTextFile("Hello", "java");
  hello.setText("public class HelloWorld{/*...*/}");
  SysTextFile test = new SysTextFile("Test", "java");
  SysTextFile prt = new SysTextFile("PrintClass", "java");
  SysDirectory srcDir = new SysDirectory("SRC", hello, test, prt);

  SysTextFile todo = new SysTextFile("Todos", "txt");
  SysTextFile toget = new SysTextFile("Eggs", "rtm");
  SysDirectory txtDir = new SysDirectory("txt", todo, toget);

  SysDirectory home = new SysDirectory("home", srcDir, txtDir);

  srcDir.setName("SRC");
  toget.setName("Einkaufsliste");
  home.setName("HOME");
  home.setOwner("brul0001");
  System.out.printf("%s\n", home );
}
```

   sollte folgende Ausgabe ergeben:

```
Class=SysDirectory, Name=HOME, User=brul0001, NumberOfObjects=2
```

Aufgabe 11.4: ***Struktur der Verzeichnisse ausgeben***

1. Ergänzen Sie bitte die Klasse `SysDirectory` um eine Instanzmethode `dirStructure`, welche die rekursive Struktur des Verzeichnisses ausgibt.
2. JavaDoc und Signatur dieser Methode:

```
/** Struktur des Verzeichnisses ermitteln
 *
 * @param indent Einrückung links (wird durch Rekursion verbreitert)
 * @return Struktur des Verzeichnisses als String
 */
      public String dirStructure(String indent)
```

3. Orientieren Sie sich bei der Lösung bitte an der Aufgabe *"Ausgeben von verschachtelten Reihungen"* und deren Methode `printInXml`.
4. Wird das `main` der letzen Teilaufgabe um

```
System.out.printf("%s\n", home );
System.out.printf("%s\n", home.dirStructure(""));
```

erweitert, so sollte die Ausgabe Folgendes sein:

```
Class=SysDirectory, Name=HOME, User=brul0001, NumberOfObjects=2
| - Class=SysDirectory, Name=SRC, User=Ulrich, NumberOfObjects=3
| - | - Class=SysTextFile, Name=Hello, User=Ulrich, Type=java, Length=32
| - | - Class=SysTextFile, Name=Test, User=Ulrich, Type=java, Length=0
| - | - Class=SysTextFile, Name=PrintClass, User=Ulrich, Type=java, Length=0
| - Class=SysDirectory, Name=txt, User=Ulrich, NumberOfObjects=2
| - | - Class=SysTextFile, Name=Todos, User=Ulrich, Type=txt, Length=0
| - | - Class=SysTextFile, Name=Einkaufsliste, User=Ulrich, Type=rtm, Length=0
```

Aufgabe 11.5: ***Information-Hiding sicherstellen***

Stellen Sie bitte für alle Ihre

- Klassen,
- Instanzvariablen und
- Instanzmethoden

sicher, dass diese durch

- Modifikatoren gegen Zugriff nach außen abgesichert werden und dass
- benötigter Zugriff nach außen nur über Getter und Setter möglich ist, falls dies wirklich gebraucht wird.

Aufgabe 11.6: ***Dokumentation***

1. Kommentieren Sie bitte alle Klassen mit JavaDoc-Kommentaren.
2. Generieren Sie die JavaDoc-Seiten.
3. Prüfen Sie die Qualität der Dokumentationsseiten.

Aufgabe 11.7: ***Struktur der Verzeichnisse durchsuchen***

Zusatzaufgabe/Knobelei:

1. Ergänzen Sie bitte die Klasse `SysDirectory` um eine Instanzmethoden `findName`, welche die rekursive Struktur des Verzeichnisses **durchsucht.**
2. JavaDoc und Signatur dieser Methode:

```
  /** Verzeichnisse nach Ressource durchsuchen, deren Namen eine
Teilzeichenkette
   *  enthält. Bei jedem Treffer wird der Name der Ressource ausgegeben.
   *
   * @param searchFor Zeichenkette nach der in den Namen der Ressourcen
gesucht wird
   *  */
  public void findName(String searchFor)
```

3. Wird das `main` der letzen Teilaufgabe um

```
String find = "e";
System.out.println("\nFind '" + find + "':");
home.findName(find);
```

erweitert, so sollte die Ausgabe Folgendes sein:

```
Find 'e':
Hello
Test
Einkaufsliste
```

13.11.2 Grafik überschreiben

(Zusatzaufgabe/Knobelei)

Aufgabe 11.8: *Sirpinski-Dreieck zeichnen*

Ziel dieser Aufgabe ist es, das Sirpinski-Dreieck am Anfang des Übungsblattes zu zeichnen.

1. Leiten Sie dazu bitte die Klasse `methods.RecuOnCanvas` zur neuen Klasse `SirpinskiOnCanvas` ab.
2. Dort brauchen Sie lediglich die Methode

   ```
   void drawRecursive(Line2D line, int depth)
   ```

   zu überschreiben. Hinweise für diese Methode:
   - In jedem Rekursionsschritt muss die im Parameter gegebene Linie gezeichnet werden.
   - Zusätzlich müssen über der Linie zwei weitere Linien gezeichnet werden, um ein gleichseitiges Dreieck zu erhalten.
   - Danach müssen noch drei Rekursionen aufgerufen werden ...