Dokumentation der `TBarcodePrinter`-Komponente

1 Anforderungen an die `TBarcodePrinter`-Komponente

Die Aufgabe der von uns geschriebenen Komponente ist, Barcodes zu erstellen und zu Drucken. Dabei soll es möglich sein, bei einem bereits teilweise bedruckten Blatt die freien Stellen zu bedrucken, um das Blatt nicht zu verschwenden. Eine wichtige Aufgabe ist auch, dass die Barcodes optisch ansprechend sind, und die Anordnung der Barcodes beim Druck in einem gewissen Rahmen veränderbar ist.
Aus der Sicht der Programmierer, die die `TBarcodePrinter`-Komponente verwenden, sollte diese einfach zu benutzen sein. Das bedeutet, die `TBarcodePrinter`-Komponente muss eine öffentliche Schnittstelle besitzen, die so groß wie nötig, aber so klein wie möglich ist. Außerdem muss in der öffentlichen Schnittstelle von Implementierungsdetails abstrahiert werden. Damit geht die Forderung einher, dass die `TBarcodePrinter`-Komponente möglichst wenig Abhängigkeiten zu anderen Komponenten besitzt, dadurch wird eine Wiederverwendung in einem anderen Kontext oder die spätere Umstrukturierung des Codes leichter ermöglicht.

2 Umsetzung der Anforderungen – Die öffentliche Schnittstelle

In einem Softwaresystem gibt es einige Ressourcen, die es nur ein einziges Mal gibt, und die zwischen verschiedenen Teilen dieses Systems geteilt werden. Dazu zählt in unserem Fall der Drucker beziehungsweise die Komponente, die Barcodes erstellt und ausdruckt – `TBarcodePrinter` . Aus diesem Grund wird `TBarcodePrinter` als Singleton realisiert, sprich es gibt nur eine einzige Instanz. Diese wird erst erstellt, wenn das erste Mal Barcodes gedruckt werden sollen. Die öffentliche Schnittstelle besteht aus folgenden fünf Methoden:

```
class function instance : TBarcodePrinter; static;
```

Zugriff auf die einzige Instanz der `TBarcodePrinter`-Komponente , gibt es noch keine Instanz, wird eine erstellt.

```
procedure add_used_sheet(n : Cardinal; used: Cardinal);
```

Das n-te Druckerblatt mehrmals verwenden, used gibt an, wie viele Stellen auf diesem Blatt schon benutzt wurden. Mehrfache Anwendung auf das selbe Druckerblatt führt zum Überschreiben der vorherigen Werte.

```
procedure add_barcode(code: String; title: String);
```

Den Barcode mit der Nummer `code`*, und der Beschriftung* `title` *zum aktuellen Druckauftrag hinzufügen;* `code` *muss ein gültiger EAN8-Code sein; Warnung: Bei langen Beschriftungen wird der letzte Teil abgeschnitten.*

```
procedure print;
```

Alle hinzugefügten Barcodes drucken.

```
procedure clear;
```

Alle eingegebenen Daten löschen.

3 Umsetzung der Anforderungen – Die Implementierung

Das Drucken von Barcodes läuft in folgenden Schritten ab. Zunächst werden in der Liste der Barcodes und in der Liste der Beschriftungen an allen notwendigen Stellen „leere Barcodes“ eingefügt. Dies passiert, wenn auf einem Blatt bereits gedruckt wurde, und die ersten Positionen nicht mehr bedruckt werden können. Leere Barcodes müssen aber auch hinzugefügt werden, wenn das letzte Blatt nicht vollständig mit echten Barcodes gefüllt werden kann, denn sonst formatiert Latex nicht richtig. Anschließend werden alle Barcodes erzeugt und in einem temporär angelegten Ordner gespeichert. In diesem Ordner wird auch die erzeugte Latex-Datei gespeichert. Ist alles vollständig, wird diese Datei kompiliert und das entstandene PDF-Dokument wird ausgedruckt. Da die erstellten Dateien (Barcodes und Überreste von der Kompilierung der Latex-Datei) nicht mehr gebraucht werden, wird der gesamte temporäre Ordner gelöscht.
Die Implementierung hängt somit „nur’ von einer Latex-Distribution, und der Datei `freetype-6.dll`, die für das Schreiben der Barcodenummer auf die Barcodegraphik verantwortlich ist, ab.

Die private Schnittstelle sieht wie folgt aus:

```
constructor create;
```

Der Konstruktor ist `private` *und kann deshalb nicht von außen aufgerufen werden. Der Konstruktor wird nur beim ersten Aufruf von* `TbarcodePrinter.instance` *aufgerufen*

```
procedure generate_latex_code;
```

Erzeugt einen den Einstellungen entsprechenden Latex-Code.

```
procedure compile_latex_code;
```

Erzeugt aus dem Latex-Code mit Hilfe von `pdflatex` *eine PDF-Datei*

```
procedure generate_barcodes;
```

Erzeugt für alle übergebenen Nummern die Barcodes

```
procedure run_command(text : string);
```

Führt das Kommando `text` *in der* `CMD` *aus.*

```
procedure fill_empty_spaces;
```

Jede Seite enthält die gleiche Zahl von Barcodes, an manchen Stellen, erscheint aber kein Barcodes. Das ist der Fall, wenn eine Stelle auf dem Druckerpapier schon benutzt wurde, oder wenn das letzte Blatt nicht ganz mit Barcodes gefüllt werden kann. Für beide Fälle müssen an die entsprechende Stelle in die Barcode- und in die Titelliste „leere Barcodes“ eingefügt werden

```
procedure create_barcode(str: string);
```

Erzeugt das Barcode-Bild für die übergebene Nummer

```
type TIntegerMap = specialize TFPGMap<Integer, Integer>;
```

Eine Map, in der die Schlüssel und die Daten vom Typ Integer sind

```
type TStringList = specialize TFPGList<String>;
```

Eine Liste von Strings

```
var unused_space_on_sheets : TIntegerMap;
```

Ordnet jedem Druckerblatt (Schlüssel) die Anzahl freier Druckerplätze (Daten) zu.

```
var titles, barcodes : TStringList;
```

Liste von Barcodes und dazugehörigen Titeln

4 Die Implementierung

Abgesehen von der Implementierung der öffentlichen und privaten Methoden werden im Implementierungsteil eine Reihe von Konstanten definiert, mit denen die Anordnung der Barcodes geändert werden kann. Außerdem wird noch eine Variable erstellt, die die einzige Instanz der Klasse `TBarcodePrinter` speichert:

```
var printer : TBarcodePrinter = nil;
```

Hier wird die einzige Instanz von TbarcodePrinter gespeichert.

```
const Tab = #9;
const NewLine = #13#10;
```

Wird zur Ausgabe von Tabs und Zeilenumbrüchen verwendet.

```
const columns = 3;
```

Anzahl der Spalten, in die das Blatt unterteilt wird

```
const column_size = 8;
```

Anzahl der Barcodes in einer Spalte

```
const page_size = columns * column_size;
```

Anzahl der Barcodes auf einer Seite

```
const barcode_width_cm = 5;
```

Breite eines Barcodes in cm

```
const barcode_height_cm = 2;
```

Höhe eines Barcodes in cm

```
const test_mode : Boolean = false;
```

Wenn `test_mode = true` *wird bei* `TbarcodePrinter.instance.print` *nichts ausgedruckt und der Ordner, in dem die PDF-Datei erstellt wird, wird nicht gelöscht. Dadurch ist es möglich, neue Einstellungen für die Anordnung der Barcodes beim Druck einfach zu testen.*

```
const working_path = '.\BarcodePrinter\';
const latex_file = 'latex.tex';
const latex_output = 'latex.pdf';
```

Dateinamen und Pfade für die verwendeten Dateien