7.3.4 Klasse PrioQueue (gb.data)

Eine Prioritäts-Queue oder PrioQueue ist eine Queue → Kapitel 7.3.3, die ihre Elemente zuerst in Prioritätsklassen einteilt und in jeder dieser Klassen der FIFO-Semantik folgt. Wie alle Klassen in der Komponente gb.data, verwendet auch PrioQueue den Datentyp *Variant* für seine Elemente.

Jedem Element kann von Ihnen bei der Einreihung in die PrioQueue eine ganze Zahl als Priorität zugewiesen werden. Je höher diese Zahl ist, desto wichtiger ist das Element und es wird weiter am Anfang der Queue eingeordnet. Besitzen zwei Elemente die gleiche Priorität – liegen in einer Prioritätsklasse – so wird das FIFO-Prinzip angewandt. Das später eingeordnete Element wird an das Ende der Prioritätsklasse angefügt.

7.3.4.1 Eigenschaften

Die Eigenschaften der PrioQueue entsprechen denen der Queue:

Eigenschaft	Datentyp	Beschreibung
.Size	Integer (ReadOnly)	Anzahl der Elemente der PrioQueue
.IsEmpty	Boolean (ReadOnly)	Wahr, wenn keine Elemente in der PrioQueue sind, sonst falsch

Tabelle 7.3.4.1.1: Eigenschaften der Klasse PrioQueue

7.3.4.2 Methoden eines Queue-Objektes

Bis auf die Enq()-Methode entsprechen die der PrioQueue denen der Queue. Die Enq()-Methode übernimmt zusätzlich die Priorität für das einzureihende Element.

Methode	Beschreibung
Clear()	Entfernen aller Elemente vom Stapel
Enq(vElement, iPrio)	Synonym für Enqueue(vElement, iPrio)
Enqueue(vElement, iPrio)	Einreihen eines neuen Elements mit der angegebenen Priorität
Deq()	Synonym für Dequeue()
Dequeue()	Entfernen und zurückgeben des ersten Elements
Peek()	Rückgabe des ersten Elements, ohne es zu entfernen

Tabelle 7.3.4.2.1: Methoden der Klasse PrioQueue

Der folgende Code:

```
Dim hPrioQueue As New PrioQueue

hPrioQueue.Enq("x", 2)
hPrioQueue.Enq("y", 1)
hPrioQueue.Enq("z", 2)

Print hPrioQueue.Deq()
Print hPrioQueue.Deq()
Print hPrioQueue.Deq()
```

erzeugt diese Ausgabe in der Konsole:

```
x
z
y
```

"x" und "z" liegen in einer Prioritätsklasse (→ 2). Da "z" später als "x" eingereiht wurde, liegt es weiter hinten in derselben Klasse. "y" mit geringerer Priorität ist das letzte Element in der Ausgabe.

7.3.4.3 Projekt zum Einsatz der Klasse PrioQueue

Der Gambas-Entwickler Bruce Bruen erwähnte, dass er die Klasse PrioQueue nutze, um 56 SQL-Skripte in 5 Prioritätsklassen zur Integritätsprüfung und Statistikerstellung für eine große Datenbank auszuführen. Bis dahin musste diese Funktionalität in einem größeren Shell-Skript implementiert und manuell gepflegt werden, was bei gelegentlich hinzuzufügenden Skripten zunehmend mühsam wurde. Seit Gambas 3.2 kann ein sehr kleines Gambas-Programm aus dem Header der Skriptdateien dynamisch die Priorität des Skripts ermitteln und alle Skripte in der vorgesehenen Reihenfolge ausführen.

Angelehnt an diesen Einsatzfall soll zur Demonstration der Klasse PrioQueue ein Projekt vorgestellt werden, das eine Vielzahl von Shell-Skripten in Prioritätsstufen unterteilt (5 Skripte in 4 Stufen werden Ihnen im Projekt zur Verfügung gestellt) und ausführt. Die Reihenfolge der Ausführung der Skripte innerhalb einer Prioritätsstufe ist beliebig → Level 3 mit 2 Skripten. Alle Skripte mit höherer Prioritätszahl müssen vor denjenigen mit geringerer Prioritätszahl ausgeführt werden – was der Semantik der PrioQueue entspricht. Die Priorität wird in jeder Skript-Datei über eine Zeile mit der Syntax:

```
# PRIO <X>
```

festgelegt, wobei <X> die Priorität [0..9] eines Skripts im *vorgestellten* Projekt ist.

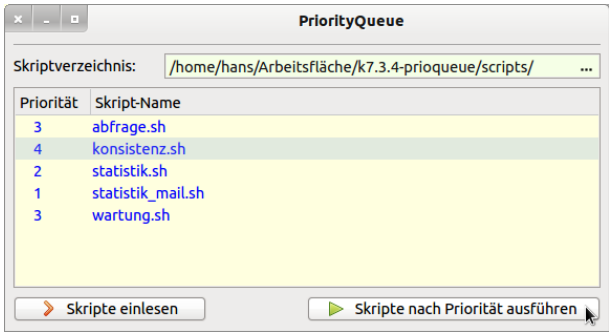

Abbildung 7.3.4.3.1: Projekt PriorityQueue (Start)

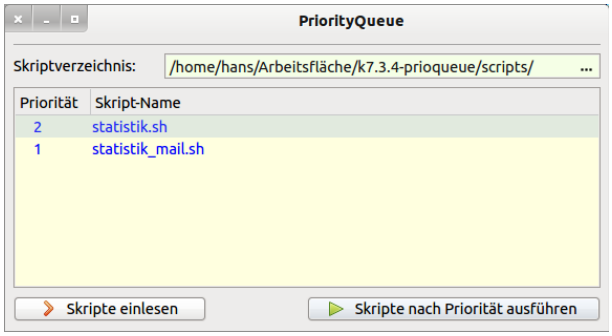

Abbildung 7.3.4.3.2: Projekt PriorityQueue (Abarbeitung Level 2)

Dieses System der Einteilung der Skripte in definierte Prioritätsstufen ist vergleichbar mit den einzelnen Runleveln, die beim Start eines Unix-Betriebssystems durchlaufen werden → http://de.wikipedia.org/wiki/Runlevel.

Eine Ausnahme müssen Sie beachten: Eine PrioQueue kann nicht rückwärts durchlaufen werden!