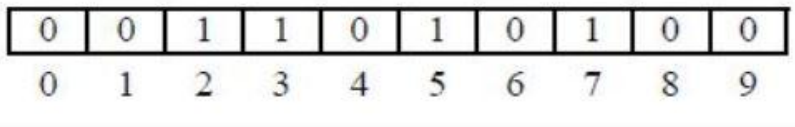
Prof. Dr. C. Yuan
Praktikum "Praktische Informatik 2" **TH Köln, Fakultät 07**

Aufgaben zu Versuch 2

Aufgabe 2.1: Eine Klasse zum Arbeiten mit Mengen

Eine Menge **M** ist bekanntlich eine Datenstruktur, die mehrere Elemente enthalten kann und in der Elemente nicht doppelt auftreten dürfen. Wenn die zugehörige Grundmenge **GM** (also die Obermenge, aus der die Elemente von **M** stammen) relativ klein ist, so lässt sich **M** durch eine „Bitmap“ darstellen: Eine Bitmap ist ein Array, in dem jedem Element der Grundmenge **GM** ein fester Index zugeordnet ist. Enthält auch M dieses Element, so steht im Array an dessen Indexposition eine **1**, sonst eine **0**.

Beispielsweise sieht für **GM = {0, 1, 2, 3, ..., 9 }** und **M = { 2, 3, 5, 7 }** die Bitmap-Darstellung wie folgt aus:



Schreiben Sie ein Java-Programm, in dem eine Klasse für solche Mengen definiert und benutzt wird. Diese Klasse soll folgende Eigenschaften haben:

- Das einzige Attribut ist ein Array zur Bitmap-Darstellung einer Menge. Sein Komponententyp soll byte sein. Die Größe des Arrays (und damit das maximale Element der Grundmenge) soll jeweils erst bei der Erzeugung eines Objekts durch den Konstruktor festgelegt werden.
- Ein Konstruktor: Der Konstruktor soll als Parameterwert die gewünschte Arraygröße übergeben bekommen. Er soll den Array erzeugen und mit Nullen vorbesetzen.
- **Drei Methoden sollen Folgendes leisten:**
  - Die **add**-Methode fügt ein neues Element in die Menge ein. Sie erhält den einzufügenden Wert als Parameter und liefert als Rückgabewert eine **-1**, wenn der Wert nicht zur Grundmenge gehört (also nicht eingefügt werden konnte), und eine **0** sonst.
  - Die **size**-Methode liefert die Größe der Menge zurück, also die Anzahl der Elemente, die zur Zeit in ihr enthalten sind (also nicht die konstante Größe der Grundmenge!).
  - Die **print**-Methode gibt die Elemente der Menge auf den Bildschirm aus.

Das Hauptprogramm soll zunächst ein Mengen-Objekt erzeugen. Dabei soll die Größe der Grundmenge (also die Länge des Arrays) über die Tastatur eingelesen und dann an den Konstruktor übergeben werden.
Anschließend soll der Benutzer die Möglichkeit haben, eine der drei Methoden auszuwählen.
Das Hauptprogramm soll, falls nötig, von der Tastatur den Parameterwert für die Methode einlesen, die Methode damit aufrufen und den Rückgabewert der Methode auf den Bildschirm ausgeben.
Dieser Vorgang soll beliebig oft ausgeführt werden, bis der Benutzer eine Beendigung wünscht.

Zu beachten:

Alle Tastatureingaben sollen im Hauptprogramm vorgenommen werden, also nicht in den Objektmethoden selbst. Auch sollen innerhalb der **add**- und der **size**-Methode keine Bildschirmausgaben erfolgen.

Aufgabe 2.2: Eine Unterklasse für Mengen mit Zusatzoperationen

Definieren Sie nun eine Unterklasse (eine abgeleitete Klasse) der Klasse aus Aufgabe 2.1.

Diese Unterklasse soll zusätzlich die folgenden Methoden bieten:

- Einen Konstruktor, der neben der gewünschten Arraygröße einen Ganzzahlwert wert übergeben bekommt und in die neu erzeugte Menge die Elemente **0, 1, ...**, **wert** einfügt. Er soll zunächst zur Erzeugung des Arrays den Konstruktor der Oberklasse aufrufen und dann selbst die Elemente in die Menge schreiben.
- Eine Methode, die genau dann **true** als Rückgabewert liefert, wenn die Menge leer ist.
- Eine erweiterte **add**-Methode, die zwei Werte unten und oben übergeben bekommt. Sie prüft zunächst, ob die beiden Werte der Grundmenge **GM** angehören. Liegt ein Wert oder liegen beide außerhalb von **GM**, so soll sofort **-1** zurückgeliefert werden. Sonst soll die Methode alle Werte **e** mit **unten < e < oben** in die Menge einfügen und eine **0** zurückgeben.

Ändern Sie Ihr Hauptprogramm so, dass ein Mengen-Objekt der Unterklasse erzeugt wird. Der Array soll die Größe **10** haben, die Grundmenge soll also **GM = { 0, 1, 2, .., 9 }** sein. Das Mengen-Objekt selbst soll mit den Elementen **{ 0, 1, 2 }** vorbesetzt werden. Wie in Aufgabe 1 soll es dann möglich sein, aus dem Hauptprogramm die Methoden der Unterklasse (einschließlich der Methoden der Oberklasse) wiederholt aufzurufen.

Aufgabe 2.3: Abstrakte Methoden

Erweitern Sie nun die Lösung aus 2.2. wie folgt:

In der Oberklasse soll eine Methode deklariert werden, die das „Mengenkomplement“ bildet, also sämtliche bisherigen Elemente aus der Menge entfernt und dafür alle Elemente der Grundmenge, die in der Menge bisher nicht enthalten waren, hinzufügt. **Beispiel: Grundmenge = {0, 1,..., 9}, Menge vorher = { 2, 3, 6, 8, 9 } ⇒ Menge nachher = Komplement der vorherigen Menge = { 0, 1, 4, 5, 7 }.**

Die Methode in der Oberklasse soll abstrakt sein, d.h. es soll ihr Kopf, aber noch nicht ihr Körper angegeben werden. Beachten Sie, dass dann auch die Oberklasse selbst als abstrakte Klasse deklariert werden muss. Erst die Unterklasse soll die Methode implementieren, d.h. einen Körper für sie definieren.

Im Hauptprogramm sollen ein Mengen-Objekt (als Objekt der Unterklasse!) erzeugt und mit einigen Werten vorbesetzt werden. Mit Hilfe der neuen Methode soll dann in diesem Mengen-Objekt das Mengenkomplement gebildet werden. Vor und nach der Komplementbildung soll der Inhalt des Mengen-Objekts auf den Bildschirm ausgegeben werden.

Aufgabe 2.4: Ausnahmebehandlung

Gehen Sie von Ihrer Lösung von Aufgabe 2.1 aus. Deklarieren Sie zunächst in Ihrem Hauptprogramm eine Objektvariable der Mengen-Klasse, weisen Sie ihr den Wert "null" zu und versuchen Sie dann, die Methode zur Bildschirmausgabe auf der Objektvariablen aufzurufen.

Welchen Laufzeitfehler erhalten Sie bei Ausführung des Programms?

Erweitern Sie nun Ihr Hauptprogramm um einen **try-catch**-Block, in dem der Fehler abgefangen wird.

Demonstrieren Sie die Funktionsfähigkeit des Programms, indem im catch-Teil die Bildschirmmeldung "**Fehler: Nullzeiger-Zugriff**" und im Anschluss an den **try-catch-**Block die Bildschirmmeldung "**Programm laeuft weiter**" ausgegeben wird.