

# Algorithmen und Datenstrukturen

## Generational Mark-Sweep-Garbage-Collector

Der Mark-Sweep-Garbage-Collector (GC) gibt automatisch nicht mehr referenzierten Speicher wieder frei. Für diesen Zweck wird zuerst in der Mark-Phase sämtlicher erreichbarer Speicher von den root-Referenzen ausgehend traversiert und markiert. Die root-Referenzen sind die statischen Variablen (Attribute) sowie die Referenzen, die sich zum Zeitpunkt des GC-Aufrufs auf dem Stack befinden. In der Sweep-Phase wird der Heap vollständig durchlaufen und nicht markierte Speicher-Blöcke bzw. Objekte werden freigegeben. Gleichzeitig wird die Markierung für den nächsten GCDurchlauf zurückgesetzt.

Für dieses Praktikum gehen wir davon aus, dass das Interface `Collectable` die Schnittstelle aller vom eigenen Garbage Collector berücksichtigten Klassen darstelle, d.h. wenn eine Klasse `Collectable` implementiert, kann sie mittels dem GC automatisch wieder freigegeben werden (in Wirklichkeit sind die Methoden und Attribute schon in `Object` definiert und nicht von Java zugänglich). Die `Collectable` implementierende Klasse muss folgende Methoden implementieren:

```
public void setMark(boolean b);  
public boolean isMarked();
```

Die Klasse `Storage` stellt die eigene Speicherverwaltung dar, wir müssen diese für diese Übung selbst übernehmen. Sie bietet die Methode `new` an, mittels der ein neues Objekt von der gegebenen Klasse instanziiert werden kann. Der Klassennamen wird als `String` übergeben sowie ein Argument für den Konstruktor. Es wird vereinfachend angenommen, dass sämtliche Objekte lediglich einen Konstruktor mit nur einem Argument vom Typ `Object` haben. Durch Aufruf der Methode `gc` wird der Garbage Collector gestartet. Die Hauptmethoden der Klasse `Storage` sind:

```
public static Collectable _new(String cls, Object arg)
```

Erzeugt neues Objekt der Klasse `cls`; der Konstruktor dieser Klasse muss ein Objekt als Argument haben.

```
public static void gc()
```

Ruft den Garbage-Collector auf.

### Aufgabe 1: Implementierung des einfachen Mark-Sweep

Zeichnen Sie die in `TestCObject` erstellte Datenstruktur auf und überlegen Sie sich, welche Objekte beim Aufruf von `gc()` Aufruf eingesammelt werden können (wir vernachlässigen in diesem Praktikum die Methode `finalize`, da diese deprecated ist).

### Aufgabe 2 Implementierung des einfachen Mark-Sweep

Vervollständigen Sie die Klasse `Storage`, so dass nicht mehr referenzierter Speicher automatisch wieder freigegeben wird, sobald die `gc()`-Methode aufgerufen wird.

Hinweise:

- Die Klasse `Cobject` ist ein Beispiel einer Klasse, die `Collectable` implementiert.
- Betrachten Sie die Methoden in der Klasse `Storage`, viele davon können Sie verwenden um in der Klasse `Storage` den Garbage-Collector zu implementieren.
- Die Klasse `TestCObject` enthält ein kleines Testprogramm für die `Storage`-Klasse. Bei korrekter Implementation des Garbage-Collectors sollten Sie folgenden Output erhalten.

ExBox connected to TestObjectServer

File Server View

New: A  
New: B  
New: C  
New: D  
New: E  
New: F  
New: G  
New: H  
  
Roots: A E  
  
Heap 1: A B C D E F G H  
  
Collector start  
Delete: H  
Collector end  
Heap 2: A B C D E F G  
  
Collector start  
Delete: C  
Delete: D  
Collector end  
Heap 3: A B E F G  
  
Collector start  
Delete: G  
Collector end  
Heap 4: A B E F  
  
Collector start  
Collector end  
Heap 5: A B E F

☒reconnect

enter

### Aufgabe 3: Garbage-Collector mit Generationen

Unser Garbage-Collector unterscheidet noch keine Generationen, das soll nun in der 3. Aufgabe integriert werden. Der Heap soll in eine youngGeneration und eine oldGeneration unterteilt werden. Der oldGeneration-Heap wird nur bei jedem 2. Aufruf bereinigt. In der Mark-Phase dürfen Sie die Differenzierung zwischen old- und newGeneration-Heap vernachlässigen (ansonsten müsste man alle Objekte im oldGeneration-Heap, die auf Objekte im newGeneration-Heap verweisen, den Wurzeln "hinzufügen"). Nach erfolgreicher Implementierung müsste Ihr Output etwa wie folgt aussehen:

```
ExBox connected to TestCObjectServer
File Server View
New: A
New: B
New: C
New: D
New: E
New: F
New: G
New: H

Roots: A E
Young heap 1: A B C D E F G H
Old heap 1:

Collector start young generation only
Delete young heap: H
Collector end
Young heap 2:
Old heap 2: A B C D E F G

Collector start young and old generation
Delete old heap: C
Delete old heap: D
Collector end
Young heap 3:
Old heap 3: A B E F G

Collector start young generation only
Collector end
Young heap 4:
Old heap 4: A B E F G

Collector start young and old generation
Delete old heap: G
Collector end
Young heap 5:
Old heap 5: A B E F

☒ reconnect  enter
```