Zürcher Hochschule für Angewandte Wissenschaften

Algorithmen und Datenstrukturen

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 GC-Durchlauf 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();
public void finalize();
```

Die Klasse `Storage` stellt die eigene Speicherverwaltung dar. Sie bietet die Methode new an, mittels der ein neues Objekt von der gegebenen Klasse instanziert 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 Objekt haben. Durch Aufruf der Methode gc wird der Garbage Collector gestartet. Die wichtigsten public Methoden 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()`	Aufruf des Garbage Collectors

Aufgabe 1 Implementierung des Mark-Sweep [7 Punkte]

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

Hinweise:

- Das `Collectable`-Interface:

```
public interface Collectable {
   public void setMark(boolean b);
   public boolean isMarked();
   public void finalize();
}
```

- `CObject` ist ein Beispiel einer Klasse, die `Collectable` implementiert.
- Folgende Hilfsmethoden aus der Klasse `Storage` können verwendet werden:

`public static Iteraterable getRoot()`	Für Iteration über alle root-Objekte.
`public static Iteraterable getHeap()`	Für Iteration über die Heap-Objekte.
`public static Iteraterable getRefs (Collectable obj)`	Für Iteration über die Referenzen auf weitere `Collectable` Objekte eines Objekts.
`public static void dump(String s, Iteraterable itr)`	Ausgabe aller Objekte
`public static String getLog()`	Log als String

- Die Methode `getRefs` bestimmt alle Referenzen auf andere CObjects
- Die Klasse `StorageServer` enthält ein kleines Testprogramm für die Storage-Klasse. Beim ersten gc-Aufruf sollten H und beim zweiten C und D freigegeben werden; Output:

```
root A E
heap1 A B C D E F G H
collect
delete H
heap2 A B C D E F G
collect
delete C
delete D
heap3 A B E F G
collect
heap4 A B E F G
collect
heap5 A B E F G
```

Abgabe

Praktikum: ADS13.1

Filename: Storage.java

Aufgabe 2 Aufruf des Finalizers [3 Punkte]

In dieser Aufgabe soll zusätzlich der Finalizer aufgerufen werden. Dabei darf das Objekt jedoch nicht sofort freigeben werden, sondern muss in einer Liste von finalisierten Objekten eingetragen werden. Erst beim nächsten gc-Aufruf kann das Objekt dann aus dem Speicher entfernt werden, sofern es in der Zwischenzeit nicht wieder referenziert wurde.

Hinweise:

- Die Liste `finalizeStage` in Storage verwenden.
- Nur die `sweep` Methode muss angepasst werden.
- Über die Methode `enableFinalizer` wird die Finalisierung in Storage aktiviert.
- Mittels der Methode `contains` kann abgefragt werden, ob ein Objekt in einer Liste enthalten ist.

```
root A E
heap1 A B C D E F G H
collect
finalize H
heap2 A B C D E F G H
collect
finalize C
finalize D
delete H
heap3 A B C D E F G
collect
delete C
delete D
heap4 A B E F G
collect
heap5 A B E F G
```

Abgabe

Praktikum: ADS13.2

Filename: Storage.java