Garbage Collectors

In this Project, We will implement three of the known algorithms used in Garbage Collectors:

- **Mark & Sweep GC**

  Algorithm: Mark phase

  ```
  Mark(root)
  If markedBit(root) = false then
      markedBit(root) = true
          For each v referenced by root
              Mark(v)
  ```

  Algorithm: Sweep Phase

  ```
  Sweep()
  For each object p in heap
  If markedBit(p) = true then
      markedBit(p) = false
  else
      heap.release(p)
  ```

- **Mark & Compact GC**

  The mark-and-compact algorithm consists of two phases:

  Phase 1: it finds and marks all live objects.

  phase 2: it is called the *mark* phase. In this phase, the garbage collection algorithm compacts the heap by moving all the live objects into contiguous memory locations.

  The algorithm can be expressed as follows:

  ```
  for each root variable r
     mark (r);
  compact ();
  ```

- **Copy GC**

  A copying garbage collector uses two heaps:

  1. from-space: the current working heap
  2. to-space: needs to be in memory during garbage collection only

At garbage collection time, the garbage collector creates an empty to-space heap in memory (of the same size as the from-space), copies the live objects from the from-space to the to-space (making sure that pointers are referring to the to-space), disposes the from-space, and finally uses the to-space as the new from-space.

The input to program:

- File **heap.csv** : it represents the information about a single allocated object. This object may be used or not used.
    - *object-identifier:* a unique 6 digits identifier of the allocated objects.
    - *memory-start:* the index of the first byte in heap memory representing this object
    - *memory-end:* the index of the last byte in heap memory representing this object

File **roots.txt**: it lists object-identifiers that are currently in use.Each line in this file contains a single object-identifier.

File **pointers.csv**: It stores the dependencies between different objects.

- *parent-identifier:* a unique identifier for the parent object
- *child-identifier:* a unique identifier for the child object referenced by the parent

The output of program:

File **new-heap.csv**: it shows the new memory layout after running the garbage collector.

Execution algorithms:

Program is executed by jar file which takes four arguments:

- The first three arguments are the absolute paths of heap.csv, roots.txt, and pointers.csv.
- The last argument is the absolute path in which the new-heap.csv file will be saved to