

Heap Heap Hooray: Memory Management

Tyler Gutowski, Trevor Schiff,
Dr. Ryan Stansifer (client)

Task	Description	Tyler	Trevor
<i>Implement memory slab system</i>	Contiguous, resizable memory chunks for use with copying garbage collection, and other algorithms that require low-level heap access.	0.3	0.7
<i>Implement copying garbage collection</i>	“Copying” garbage collection algorithm	0.6	0.4
<i>Design compiler flag/configuration user interface</i>	Design compiler user-interface, including flags that can be used when invoking the compiler (--gc, --opt, --verbose, etc.)	0.8	0.2
<i>Implement compiler flag/configuration functionality</i>	Implement the required functionality behind the compiler flags	0.2	0.8
<i>Implement compiler flag for setting garbage collection method</i>	Bake the garbage collection type into the executable, where the C runtime can see and then toggle the appropriate features	0.0	1.0
<i>Run tests across refcount/markswEEP/copying garbage collection methods</i>	Create and run thorough test cases across all garbage collection implementations	N/A	N/A
<i>Gather metrics for refcount/markswEEP/copying garbage collection methods</i>	Analyze test case results and gather further metrics	N/A	N/A

Basic Overview: Copying

When we try to allocate memory, but none is available, perform:

1. Mark Phase

- a. Same functionality as the mark phase of mark-sweep GC
- b. Traverse object graph & mark all reachable allocations
- c. Start from the program roots, or objects reachable from outside the heap (such as local variables)

2. Copy Phase

- a. All memory allocations exist in one of two “slabs” (contiguous chunks) of memory: the “from” slab, or the “to” slab
 - i. “From” slab contains all allocations
- b. Copy all live allocations from the “from” slab to the “to” slab
 - i. “To” slab contains only live (reachable) allocations
- c. Remove allocations from the “from” slab that have just been copied
 - i. “From” slab contains only garbage (unreachable) allocations
- d. Release allocations which still exist in the “from” slab
 - i. All memory associated with garbage has been released
- e. Swap handles to the “from” and “to” slabs
 - i. “From” slab contains only live (reachable allocations)

Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph

Main\$main

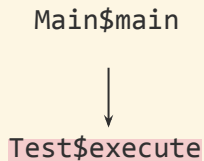
|

Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph

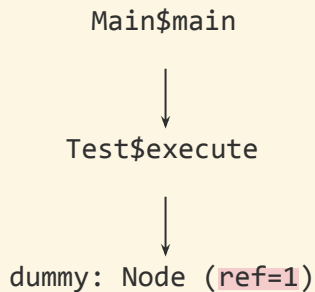


Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph



Implementation: Copying

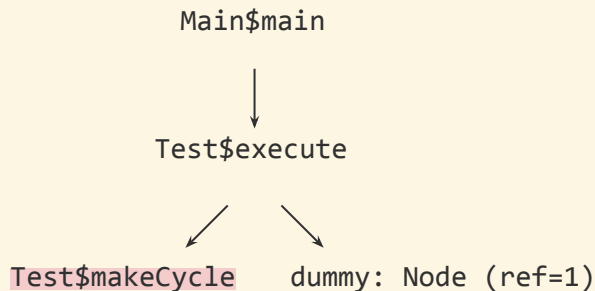
Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}
```

```
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
}
```

```
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph



Implementation: Copying

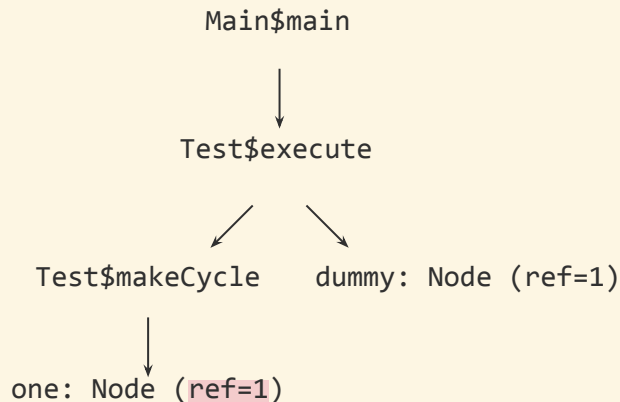
Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}
```

```
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
}
```

```
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph

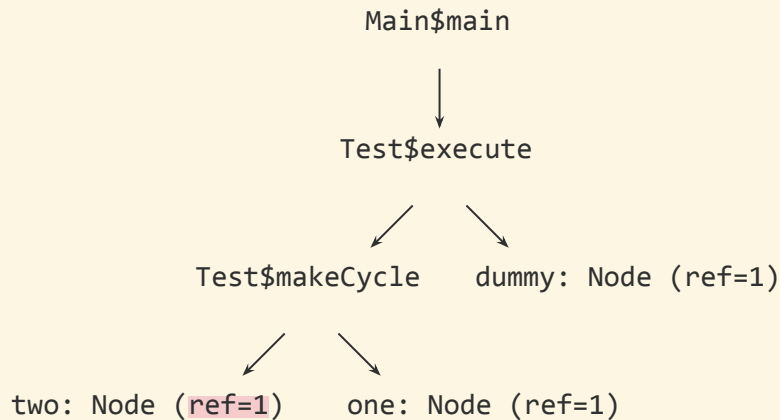


Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph

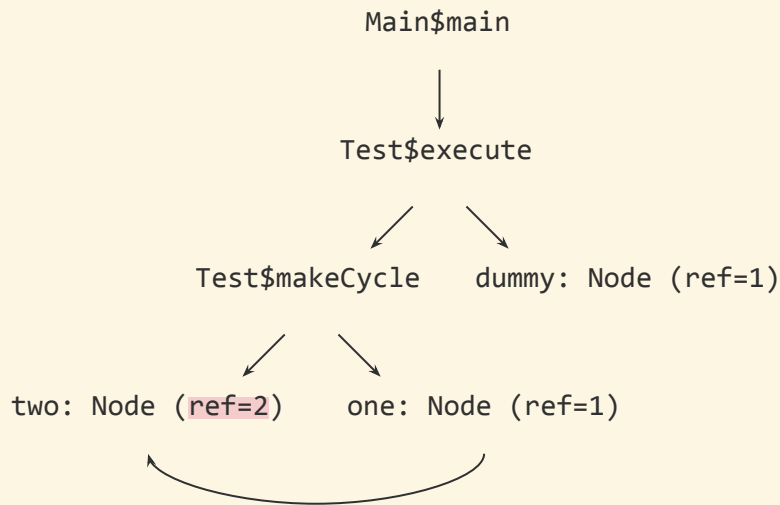


Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph



Implementation: Copying

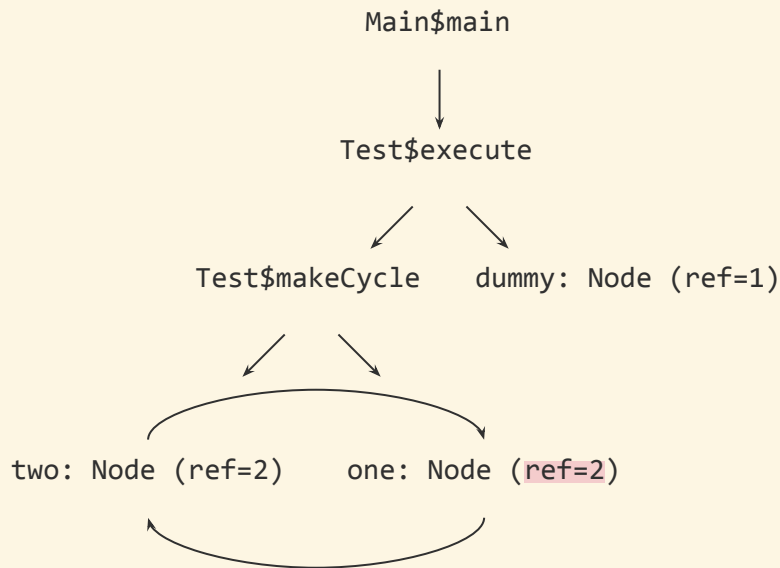
Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}
```

```
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
}
```

```
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph

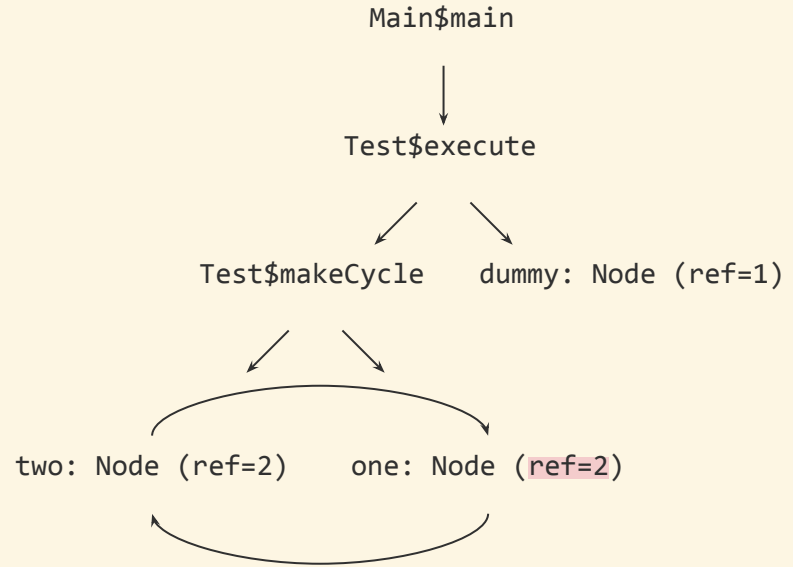


Implementation: Copying

Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}  
  
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
  
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph



← End of scope, decrement
refcount of 'one' & 'two'

Implementation: Copying

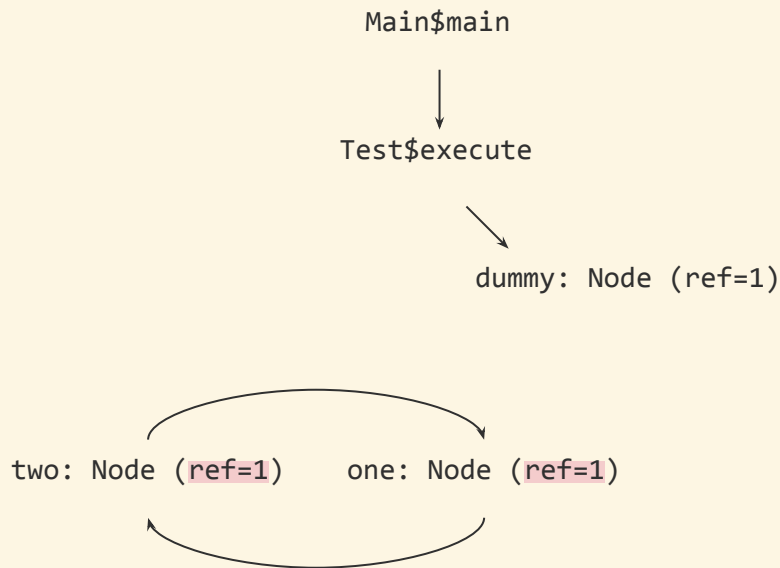
Program

```
class Main {  
    public static void main(String[] a) {  
        new Test().execute();  
    }  
}
```

```
class Test {  
    public int execute() {  
        Node dummy = new Node();  
        makeCycle();  
    }  
}
```

```
    public void makeCycle() {  
        Node one = new Node();  
        Node two = new Node();  
  
        one.setNext(two);  
        two.setNext(one);  
    }  
}
```

Graph



Not enough, they cannot be freed!
Memory will be leaked!

Implementation: Copying

Graph

Begin “mark” phase.

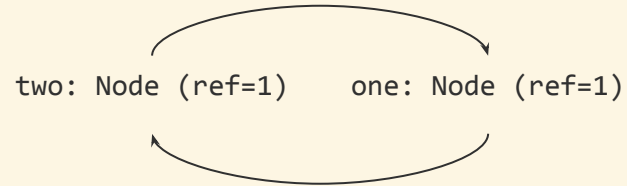
Main\$main



Test\$execute



dummy: Node (ref=1)



Implementation: Copying

Graph

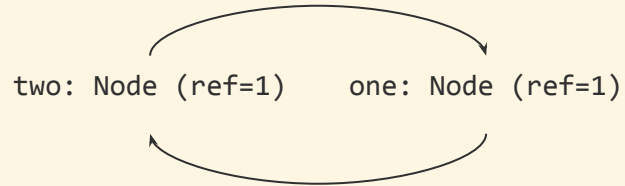
Main\$main



Test\$execute



dummy: Node (ref=1)



Implementation: Copying

Graph

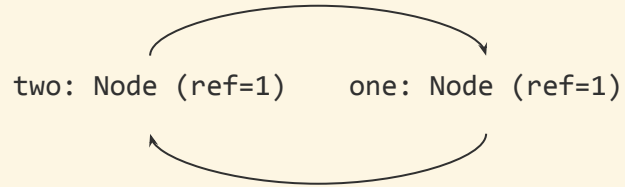
Main\$main



Test\$execute



dummy: Node (ref=1)



Implementation: Copying

Graph

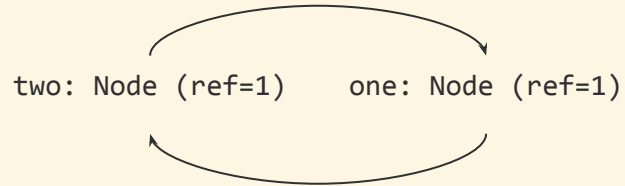
Main\$main



Test\$execute



dummy: Node (ref=1)



Implementation: Copying

Graph

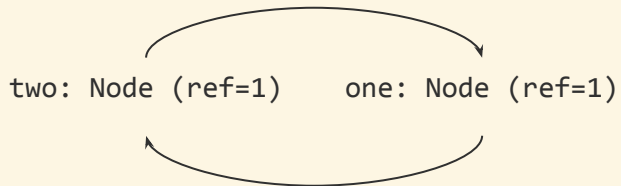
Main\$main



Test\$execute



dummy: Node (ref=1)



These will never be marked!

Implementation: Copying

“From” slab

Begin “copy” phase.

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>one</i>	0x25068	0x000C	No
<i>two</i>	0x25088	0x000C	No
<i>(free space)</i>	N/A	0xFFDB	N/A

“To” slab

Name	Address	Size	Marked?
<i>(free space)</i>	N/A	0xFFFF	N/A

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>one</i>	0x25068	0x000C	No
<i>two</i>	0x25088	0x000C	No
<i>(free space)</i>	N/A	0xFFDB	N/A

Copy all live
allocations to the
“to” slab

→ OK

→ X

→ X

“To” slab

Name	Address	Size	Marked?
<i>(free space)</i>	N/A	0xFFFF	N/A

Implementation: Copying

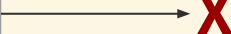
“From” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>one</i>	0x25068	0x000C	No
<i>two</i>	0x25088	0x000C	No
<i>(free space)</i>	N/A	0xFFDB	N/A

Copy all live
allocations to the
“to” slab

“To” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>(free space)</i>	N/A	0xFFF3	N/A



Implementation: Copying

“From” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>one</i>	0x25068	0x000C	No
<i>two</i>	0x25088	0x000C	No
<i>(free space)</i>	N/A	0xFFDB	N/A

Release allocations
in the “from” slab
that were just
copied

“To” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>(free space)</i>	N/A	0xFFFF3	N/A

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
<i>one</i>	0x25068	0x000C	No
<i>two</i>	0x25088	0x000C	No
<i>(free space)</i>	N/A	0xFFE7	N/A

Now only garbage

Release allocations
in the “from” slab
that were just
copied

“To” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>(free space)</i>	N/A	0xFFF3	N/A

Now only non-garbage

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
one	0x25068	0x000C	No
two	0x25088	0x000C	No
(free space)	N/A	0xFFE7	N/A

Release allocations
which still exist
in the “from” slab

“To” slab

Name	Address	Size	Marked?
dummy	0x25028	0x000C	Yes
(free space)	N/A	0xFFF3	N/A

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
<i>(free space)</i>	N/A	0xFFFF	N/A

All garbage collected!

Release allocations which still exist in the “from” slab

“To” slab

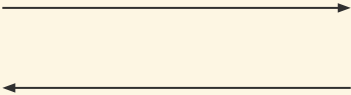
Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>(free space)</i>	N/A	0xFFF3	N/A

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
(free space)	N/A	0xFFFF	N/A

Swap handles to the
“from” and “to”
slabs



“To” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
(free space)	N/A	0xFFF3	N/A

Implementation: Copying

“From” slab

Name	Address	Size	Marked?
<i>dummy</i>	0x25028	0x000C	Yes
<i>(free space)</i>	N/A	0xFFF3	N/A

Garbage collection
cycle is finished.

“To” slab

Name	Address	Size	Marked?
<i>(free space)</i>	N/A	0xFFFF	N/A

Demo: Copying

```
debug > mv tmpfile debug; chmod u+w debug; tr -d '\r' < build > tmpfile && mv tmpfile build; chmod u+w build;
id; tr -d '\r' < Makefile > tmpfile && mv tmpfile Makefile; chmod u+w Makefile;
Job Interact rt
Linux debian 5.10.0-18-amd64 #1 SMP Debian 5.10.140-1 (2022-08-02) x86_64

=====
CSE 4251/5251: Compiler Construction (Spring 2023)

All included SPARC GNU tools are prefixed with 'sparc-linux-'.
Notable examples include:
- sparc-linux-gcc
- sparc-linux-as
- sparc-linux-ld
- sparc-linux-gdb
- etc.

All included MIPSel GNU tools are prefixed with 'mipsel-linux-'.
The MIPSel package contains the same tools as the SPARC package.

This container is shipped with OpenJDK 17.

This program is still in early development and will have a lot of bugs.
Please report any issues to https://github.com/Klippliz/Jabberwocky/Issues.
This program is distributed with ABSOLUTELY NO WARRANTY.
=====

Last login: Mon Feb 19 12:49:27 2024 from 10.0.2.1
root@debian:~# ./build CyclinGarbageTest --gcc=copying
tests/CyclinGarbageTest.java
```

Implementation: Copying

```
root@debian:~# ./CyclicGarbageTest
[runtime/config.c:42] setting gctype to Copying
[runtime/markswep.c:52] push_stack 0x40800660 (size:96)
[runtime/heap.c:143] alloc 0x25040 (size:4), userptr: 0x25048
[runtime/markswep.c:52] push_stack 0x408005f8 (size:100)
[runtime/heap.c:143] alloc 0x2504c (size:4), userptr: 0x25054
[runtime/heap.c:143] alloc 0x25058 (size:4), userptr: 0x25060
[runtime/markswep.c:52] push_stack 0x40800598 (size:92)
[runtime/markswep.c:70] pop_stack
[runtime/markswep.c:52] push_stack 0x40800598 (size:92)
[runtime/markswep.c:70] pop_stack
[runtime/markswep.c:70] pop_stack
[runtime/markswep.c:144] search stack frame: 0x40800660
(size:96)
```

```
class CyclicGarbageTest {
    public int execute() {
        Node dummy;

        dummy = new Node();

        this.leak();
        System.gc();

        return 0;
    }

    public int leak() {
        Node one;
        Node two;

        one = new Node();
        two = new Node();

        one.setNext(two);
        two.setNext(one);

        return 0;
    }
}
```

Implementation: Copying

```
[runtime/markswEEP.c:148] sp->lreg[0]=408006BC
[runtime/markswEEP.c:148] sp->lreg[1]=00000000
[runtime/markswEEP.c:148] sp->lreg[2]=00000000
[runtime/markswEEP.c:148] sp->lreg[3]=00000000
[runtime/markswEEP.c:148] sp->lreg[4]=00000000
[runtime/markswEEP.c:148] sp->lreg[5]=00000000
[runtime/markswEEP.c:148] sp->lreg[6]=00000000
[runtime/markswEEP.c:148] sp->lreg[7]=000240C8
[runtime/markswEEP.c:155] sp->ioreg[0]=408006BC
[runtime/markswEEP.c:155] sp->ioreg[1]=00000000
[runtime/markswEEP.c:155] sp->ioreg[2]=00000000
[runtime/markswEEP.c:155] sp->ioreg[3]=00000000
[runtime/markswEEP.c:155] sp->ioreg[4]=00000000
[runtime/markswEEP.c:155] sp->ioreg[5]=00000000
[runtime/markswEEP.c:162] local_num=1 ← found dummy
[runtime/markswEEP.c:192] sp->locals[0 (align:0)] = 00025048
[runtime/markswEEP.c:103] mark 0x25040
[runtime/markswEEP.c:107] search allocated block 0x25048
[runtime/copying.c:126] copying 0x25040 from the "from" slab ← copying dummy
[runtime/heap.c:143] alloc 0x35080 (size:4), userptr: 0x35088 ← for copy
[runtime/slab.c:62] destroy: freeing something (0x450a8, begin->0x2504c)
[runtime/slab.c:62] destroy: freeing something (0x450f8, begin->0x25058)
[runtime/markswEEP.c:70] pop_stack
[runtime/heap.c:101] allocated:
[runtime/heap.c:105] addr:0x35080 size:4 ref:0 ← dummy was preserved
```

```
class CyclicGarbageTest {
    public int execute() {
        Node dummy;

        dummy = new Node();

        this.leak();
        System.gc();

        return 0;
    }

    public int leak() {
        Node one;
        Node two;

        one = new Node();
        two = new Node();

        one.setNext(two);
        two.setNext(one);

        return 0;
    }
}
```

Milestone 5 Goals

- Implement “generational” GC method
 - Relies on copying GC
 - Improves efficiency of GC cycles by less frequently trying to collect objects that could not previously be freed
- Conduct evaluation and analyze results
 - Compare implementations
 - Gather metrics and meaningful conclusions
- Create poster and ebook page for Senior Design Showcase
 - Begin preparing display material for the showcase