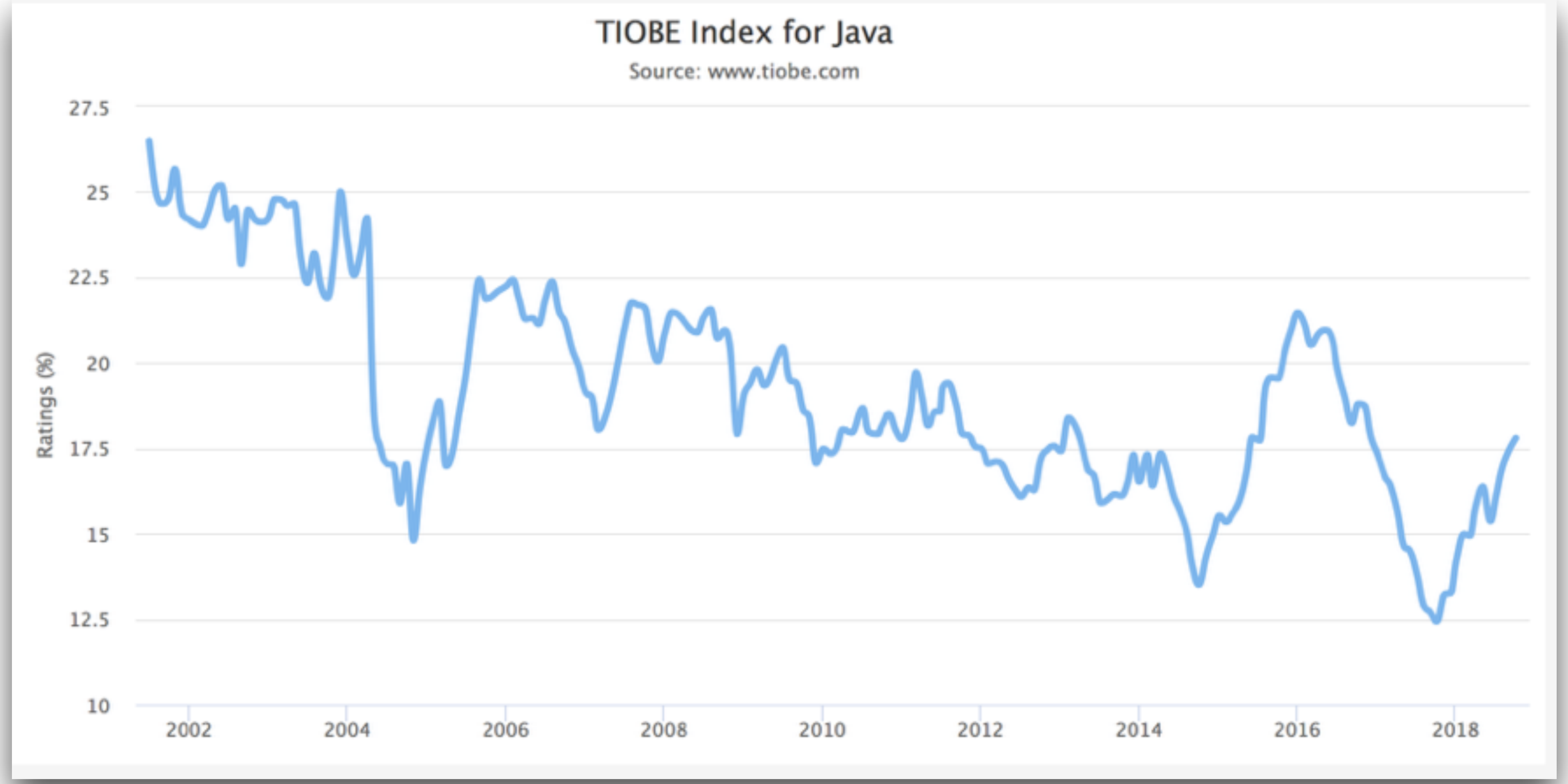
Föreläsning 16

Tobias Wrigstad

*Imperativ och **objekt-orienterad** programmering*

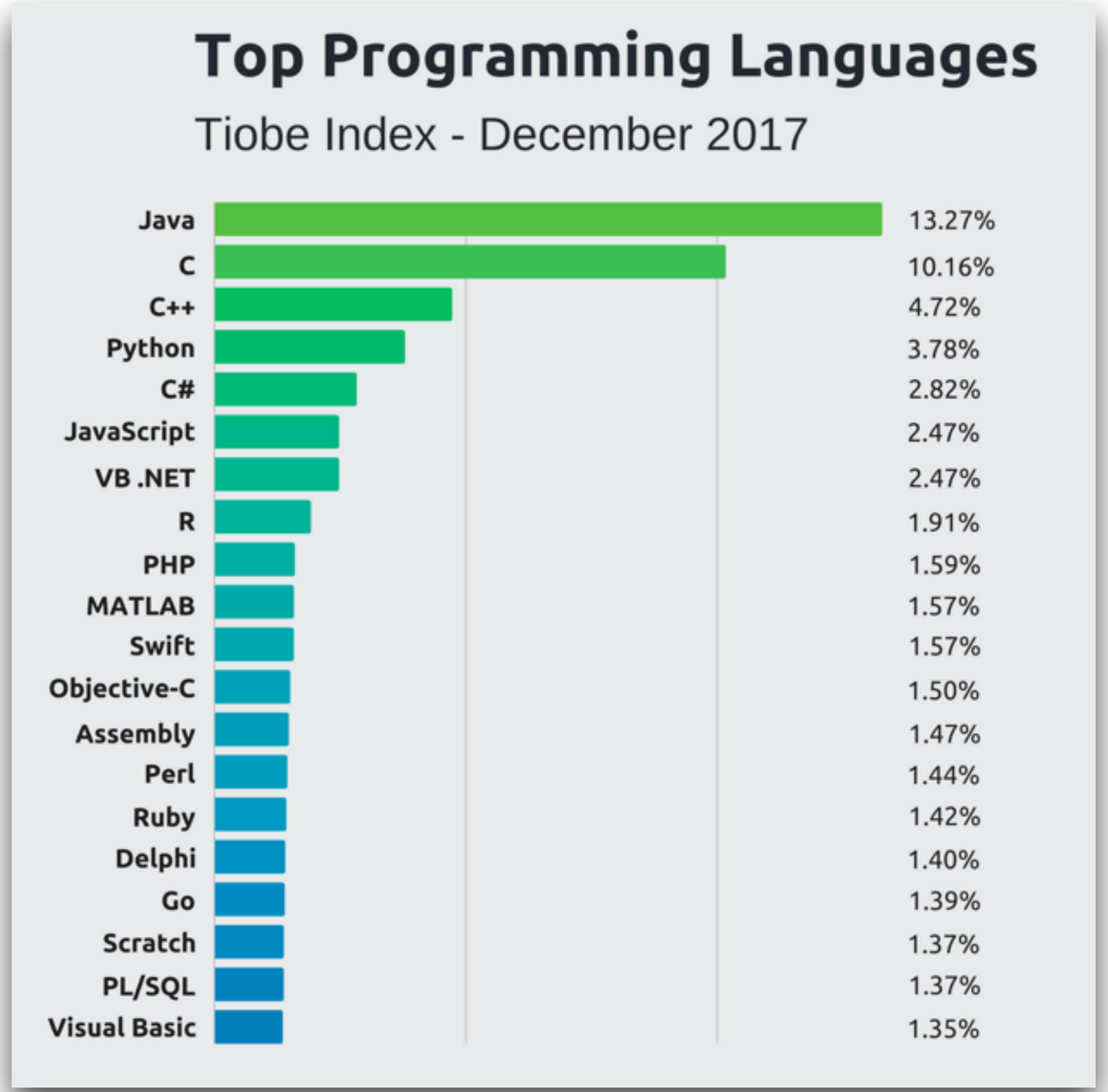
Vem använder Java?



Highest Position (since 2001): #1 in Oct 2018

Lowest Position (since 2001): #2 in Mar 2015

Källa: Tiobe


Top Programming Languages
Tiobe Index - December 2017
Java 13.27%
C 10.16%
C++ 4.72%
Python 3.78%
C# 2.82%
JavaScript 2.47%
VB .NET 2.47%
R 1.91%
PHP 1.59%
MATLAB 1.57%
Swift 1.57%
Objective-C 1.50%
Assembly 1.47%
Perl 1.44%
Ruby 1.42%
Delphi 1.40%
Go 1.39%
Scratch 1.37%
PL/SQL 1.37%
Visual Basic 1.35%

Källa: Tiobe

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2017	2012	2007	2002	1997	1992	1987
Java	1	2	1	1	15	-	-
C	2	1	2	2	1	1	1
C++	3	3	3	3	2	2	4
C#	4	5	7	11	-	-	-
Python	5	7	6	12	27	16	-
Visual Basic .NET	6	14	-	-	-	-	-
JavaScript	7	9	8	7	20	-	-
PHP	8	6	4	5	-	-	-
Perl	9	8	5	4	3	8	-
Delphi/Object Pascal	10	11	11	8	-	-	-
Lisp	31	12	15	13	8	4	2
Prolog	32	30	26	15	17	13	3

Källa: Tiobe

Java är plattformsoberoende

- Datatyper har en standardstorlek
- Samtliga klasser i standardbiblioteket finns på alla maskiner
- Javas minnesmodell är densamma på alla maskiner
- Ditt program kommer att bete sig likadant på din kompis dator

  Du behöver inte ens kompilera om programmet — du kan flytta det rakt av

  Caveat: Gränssnittsprogram och OS-specifika tjänster

  `C:\foo\bar.txt` vs. `/foo/bar.txt`

Java är plattformsoberoende

Type	Default	Size	Example Literals
boolea	false	1 bit	`true`, `false`
byte	0	8 bits	(none)
char	`\u0000`	16 bits	`'a'`, `'\u0041'`,
short	0	16 bits	(none)
int	0	32 bits	`-2`, `-1`, `0`, `1`, `2`
long	0	64 bits	`-2L`, `-1L`, `0L`, `1L`, `2L`
float	0.0	32 bits	`1.23e100f`,
double	0.0	64 bits	`1.23456e300d`,

Datatyperna är samma oavsett plattform

Automatisk minneshantering

Objects know their size
(but we may not!)

```
new LinkedList();
```

Unreachable objects
are reclaimed

```
/// This program does not leak
LinkedList list = new LinkedList();
for (int i = 0; i < 1000000; ++i) {
    list.add(new Object());
}
list = null;
```

Unused objects
are not…

```
/// This program does might "leak"
LinkedList list = new LinkedList();
for (int i = 0; i < 1000000; ++i) {
    list.add(new Object());
}
```

Metadata

- En objekt känner sitt ursprung

```
Object o = new Person();

o instanceof Person // true

Class c = o.getClass();

c.newInstance(); // skapa en ny person
```

- Reflection och introspection

```
Method m = o.getClass().getMethod("setName", String.class);

ms.invoke(o, "Barbara")

for (Method m : c.getMethods()) { if (m.startsWith("test")) m.invoke(); }
```

- …and more, e.g., `array.length`

Inkapsling

- Namnbaserad inkapsling styr vem som får nämna ett visst namn
- Node är bara en valid typ inuti LinkedList
- Kräver aktivt ställningstagande från dig!
- Standard är ”package” — dvs. åtkomligt överallt från modulen

```
public class Pair {
    private Object fst;
    private Object snd;
    Object getFst() { return this.fst; }
    void setFst(Object o) { this.fst = o; }
}
```

```
public class LinkedList {
    private Node first = new Node();
    private class Node {
        Node next;
        Object element;
        public Node(Object o, Node n) {
            this.element = o;
            this.next = n;
        }
    public void prepend(Object o) {
        this.first =
            new Node(o, this.first);
    }
…
```

Världens rikaste standardbibliotek(?)

- Sök på ”java 10 api KlassensNamn”
- Genererat med JavaDoc utifrån kommentarer i källkoden — inspiration för D9
- Indelat i paket; viktigaste paket för er:

  `java.lang`

  Grundläggande objekt, och systemobjekt

  `java.util`

  Vanliga datastrukturer, StringTokenizer

  `java.io`

  I/O

http://docs.oracle.com/javase/X/docs/api/

docs.oracle.com

ioopm17/f6.pdf at master · IOOPM-UU/ioop... | Imperative & Object-Oriented Programmin... | HashMap (Java Platform SE 7)

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.util

Class HashMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>

Direct Known Subclasses:

LinkedHashMap, PrinterStateReasons

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
```

Hash table based implementation of the `Map` interface. This implementation provides all of the optional map operations, and permits `null` values and the `null` key. (The `HashMap` class is roughly equivalent to `Hashtable`, except that it is unsynchronized and permits nulls.) This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time.

docs.oracle.com

ioopm17/f6.pdf at master · IOOPM-UU/ioop... | Imperative & Object-Oriented Programmin... | HashMap (Java Platform SE 7)

Constructor Summary

Constructors

Constructor and Description
`HashMap()` Constructs an empty `HashMap` with the default initial capacity (16) and the default load factor (0.75).
`HashMap(int initialCapacity)` Constructs an empty `HashMap` with the specified initial capacity and the default load factor (0.75).
`HashMap(int initialCapacity, float loadFactor)` Constructs an empty `HashMap` with the specified initial capacity and load factor.
`HashMap(Map<? extends K,? extends V> m)` Constructs a new `HashMap` with the same mappings as the specified `Map`.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`clear()` Removes all of the mappings from this map.
`Object`	`clone()` Returns a shallow copy of this `HashMap` instance: the keys and values themselves are not cloned.

docs.oracle.com

ioopm17/f6.pdf at master · IOOPM-UU/ioop... | Imperative & Object-Oriented Programmin... | HashMap (Java Platform SE 7)

Method Summary

Methods

Modifier and Type	Method and Description
void	`clear()` Removes all of the mappings from this map.
Object	`clone()` Returns a shallow copy of this `HashMap` instance: the keys and values themselves are not cloned.
boolean	`containsKey(Object key)` Returns `true` if this map contains a mapping for the specified key.
boolean	`containsValue(Object value)` Returns `true` if this map maps one or more keys to the specified value.
Set<Map.Entry<K,V>>	`entrySet()` Returns a `Set` view of the mappings contained in this map.
V	`get(Object key)` Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.
boolean	`isEmpty()` Returns `true` if this map contains no key-value mappings.
Set<K>	`keySet()` Returns a `Set` view of the keys contained in this map.
V	`put(K key, V value)` Associates the specified value with the specified

docs.oracle.com

ioopm17/f6.pdf at master · IOOPM-UU/ioop... | Imperative & Object-Oriented Programmin... | HashMap (Java Platform SE 7)

put

```
public V put(K key,
    V value)
```

Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.

Specified by:

`put` in interface `Map<K,V>`

Overrides:

`put` in class `AbstractMap<K,V>`

Parameters:

`key` - key with which the specified value is to be associated

`value` - value to be associated with the specified key

Returns:

the previous value associated with `key`, or `null` if there was no mapping for `key`. (A `null` return can also indicate that the map previously associated `null` with `key`.)

putAll

```
public void putAll(Map<? extends K,? extends V> m)
```

Copies all of the mappings from the specified map to this map. These mappings will replace any mappings that this map had for any of the keys currently in the specified map.

Specified by:

Hundratals paket

Tusentals klasser

docs.oracle.com

ioopm17/f6.pdf at master · IOOPM-UU/ioop... | Imperative & Object-Oriented Programmin... | HashMap (Java Platform SE 7)

java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio

MidiDevice.Info
MidiDeviceProvider
MidiDeviceReceiver
MidiDeviceTransmitter
MidiEvent
MidiFileFormat
MidiFileReader
MidiFileWriter
MidiMessage
MidiSystem
MidiUnavailableException
MimeHeader
MimeHeaders
MimeUtility
MimeTypeParameterList
MimeTypeParseException
MimeTypeParseException
MimetypesFileTypeMap
MinimalHTMLWriter
MirroredTypeException
MirroredTypesException
MissingFormatArgumentException
MissingFormatWidthException
MissingResourceException
Mixer
Mixer.Info
MixerProvider
MLet

Overview Package Class Use Tree Deprecated

Prev Class Next Class

Frames No Frames

Summary: Nested | Field | Constr | Method
Detail: Field | Constr | Method

java.util

Class HashMap<K,V>

java.lang.Object
  java.util.AbstractMap<K,V>
    java.util.HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>

Direct Known Subclasses:

LinkedHashMap, PrinterStateReasons

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Se
```

Hash table based implementation of the Map interface. This implementation provides all of

Stark typning

- Java är starkt typat (C *svagt*!)

  Vi kan inte behandla en typ som en annan

  Försök att göra så genererar ett tydligt fel under körning

```
/// Type punning in C
elem_t e; /// unknown content
e.int_value = 42;
float f = e.float_value; ///???
```

```
/// Type casting in C is abusive
void *ptr = (void *)42;
int i = (int) ptr;
```

```
/// Bad type cast generates runtime error
Object o = new Object();
Person p = (Person) o; /// Compiles
```

ClassCastException

Parametrisk Polymorfism

```
/// Revisiting previous examples – and improving them!
Person p1 = new Person();
Class<Person> cp = p1.getClass();
Person p2 = cp.newInstance();
```

```
/// Revisiting previous examples – and improving them!
LinkedList<Person> list = new LinkedList<>(); /// !!
list.add(new Object()); /// Will not compile
Person p = list.first();
```

```
/// Revisiting previous examples – and improving them!
public class Person implements Comparable<Person> {
    ...
}
```

Inga Segfaults

Avrefererad `null`-pekare i Java:

```
Exception in thread "main" java.lang.NullPointerException
        at com.example.myproject.Book.getTitle(Book.java:16)
        at com.example.myproject.Author.getBookTitles(Author.java:25)
        at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
```

Uppslagning i array med `index < 0` eller `index >= array.length`

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
        at com.example.myproject.BookStore.getBook(BookStore.java:52)
        at com.example.myproject.BookStore.getLatest(BookStore.java:33)
        at com.example.myproject.Bootstrap.main(Bootstrap.java:17)
```

Notera att programmet skriver ut vilken rad det kraschade på!

Exception Handling

```
/// Bad type cast generates runtime error
BufferedReader in = null;
try {
    in = new BufferedReader(new FileReader("foo.in"));
    while (true) {
        node = node->next;
    }
    ...
} catch (NullPointerException e) {
    /// Went to far in the list!
    e.printStackTrace(System.err);
} finally {
    if (in != null) {
        in.close();
    }
}
```

Referenser

- Ingen pekararitmetik
- En referens kan inte skapas ur tomma intet
- Inga dangling pointers
- En **pekare är en adress** — ett heltal — ett offset från 0
- En **referens är ett handtag**, en token genom vars försorg jag kan accessa ett objekt
- Ofta säger vi pekare ändå, att de är referenser är uppenbart av kontexten
- En `null`-pekare är inte någon referens, utan är **avsaknaden av en referens**

Jshell

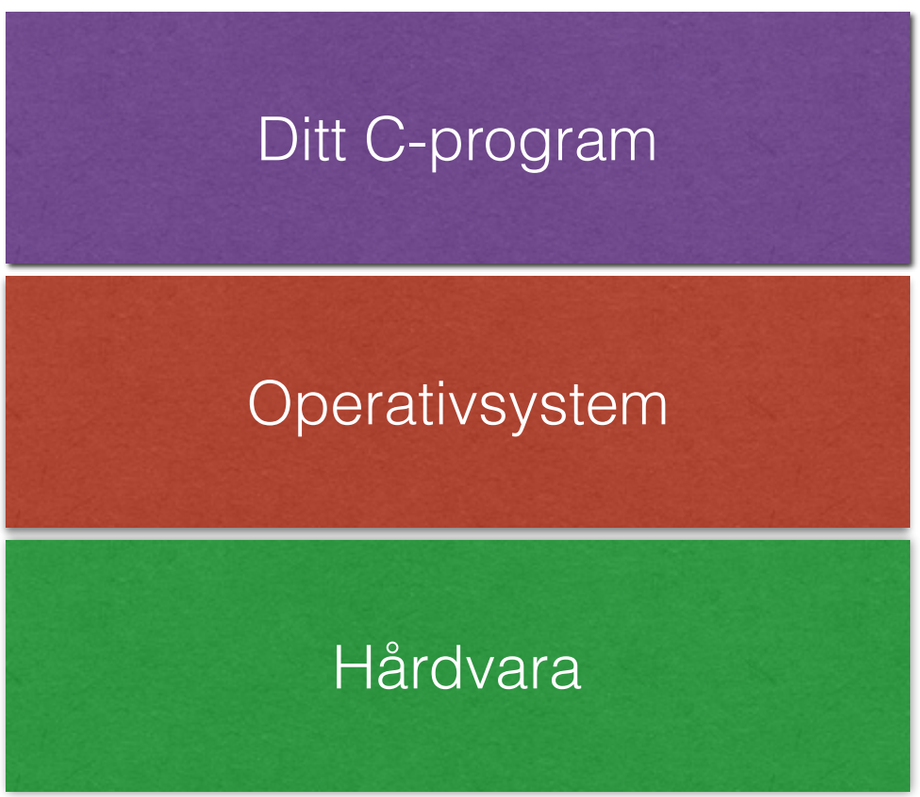
- Från och med JDK 9 har Java äntligen fått en REPL (Read-Eval-Print Loop)
- Lek med Java i en interaktiv, levande miljö

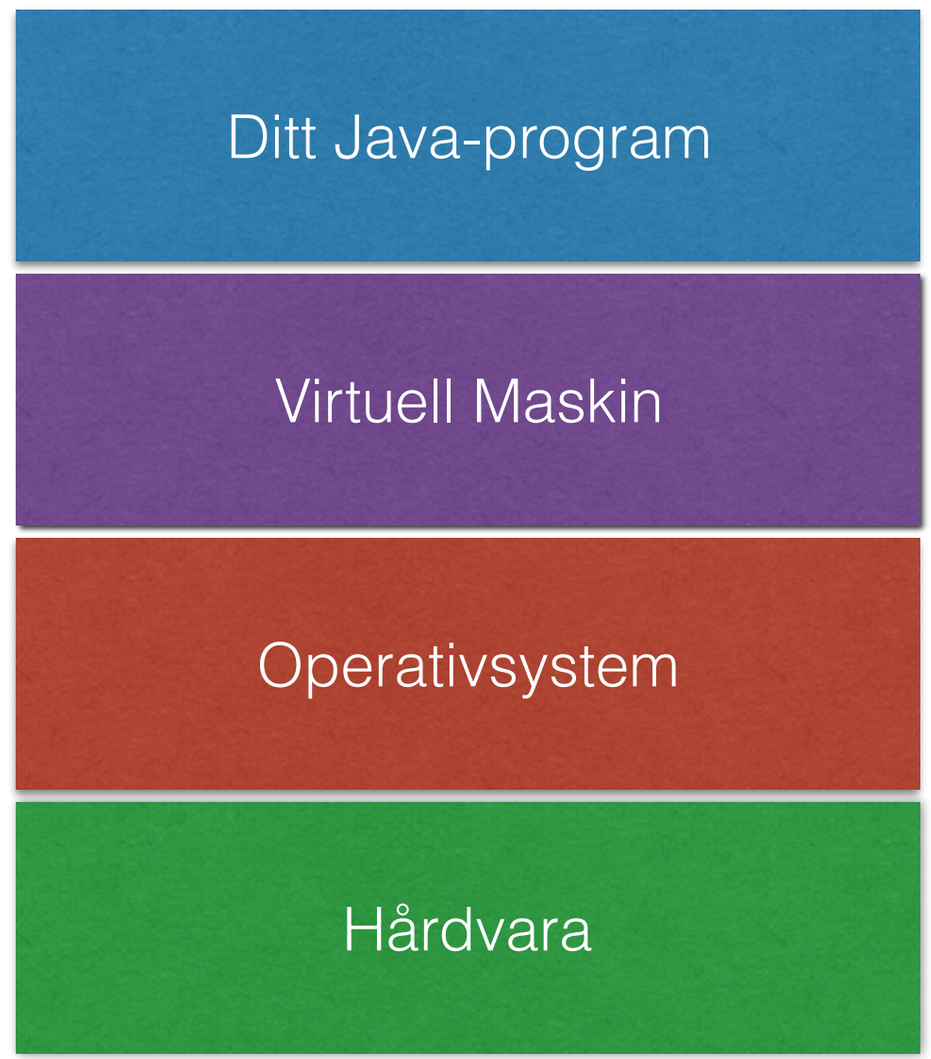
```
[writo649@trygger:6 ~]$ jshell
|  Welcome to JShell -- Version 10.0.2
|  For an introduction type: /help intro

jshell> 42 + 42
$1 ==> 84

jshell> public class Test { publict Test(int i) { this.i = i; } int i; }
|  Error:
|  cannot find symbol
|    symbol:   class publict
|  public class Test { publict Test(int i) { this.i = i; } int i; }
|                      ^-----^
|  Error:
|  missing return statement
|  public class Test { publict Test(int i) { this.i = i; } int i; }
|                                          ^-------------^

jshell> public class Test { public Test(int i) { this.i = i; } int i; }
|  created class Test

jshell> Test t = new Test()
|  Error:
|  constructor Test in class Test cannot be applied to given types;
|    required: int
|    found: no arguments
|    reason: actual and formal argument lists differ in length
|  Test t = new Test();
|           ^----^

jshell> Test t = new Test(42)
t ==> Test@5e3a8624
```

Objektorientering?

- Förhoppningsvis kommer du också att uppskatta objektorientering…

Stacken i Java vs. Stacken i C





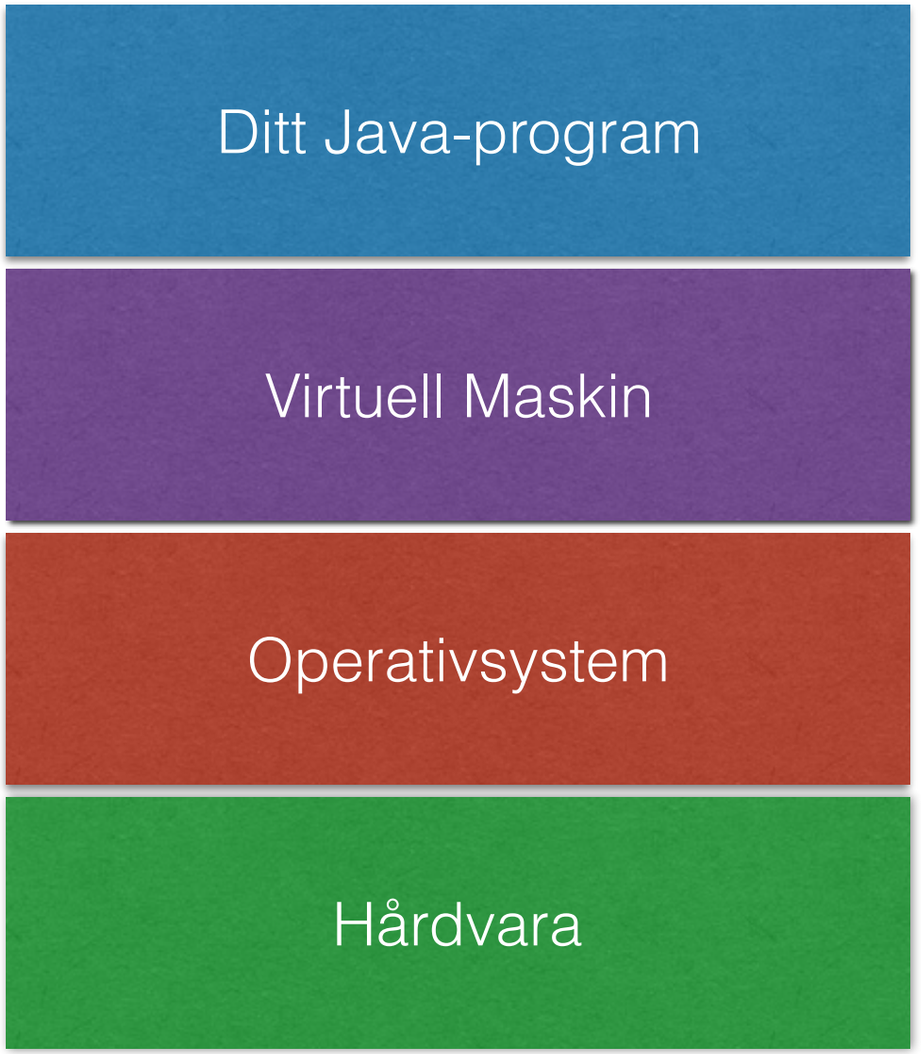
Kompilera och köra ett Java-program

```
$ javac MyProg.java
```

Skapar en eller flera .class-filer varav en heter MyProg.class

```
$ java MyProg
```

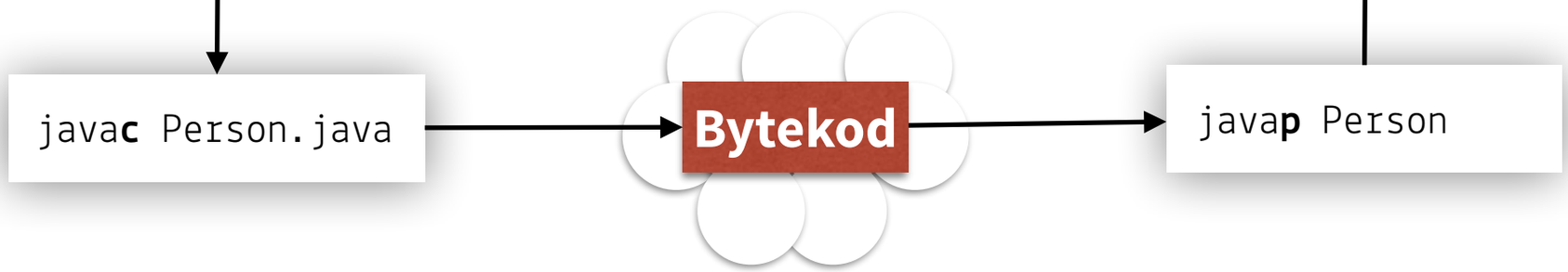
Startar den virtuella maskinen och laddar in MyProg och kör



Kompilera ditt Java-program

```java
public class Person {
    public String name;
    public Person(String name) {
     assert name != null : "Name == null!";
     this.name = name;
    }
    public void setName(String name) {
     this.name = name;
    }
    public String getName() {
     return this.name;
    }
    public String toString() {
     return "Person(" + this.name + ")";
    }
}
```

```java
public class Person {
  public java.lang.String name;
  static final boolean $assertionsDisabled;
  public Person(java.lang.String);
  public void setName(java.lang.String);
  public java.lang.String getName();
  public java.lang.String toString();
  static {};
}
```



Use the source, Luke!

```
javap -c Person
```

```
Compiled from "Person.java"
public class Person {
  public java.lang.String name;

  static final boolean $assertionsDisabled;

  public Person(java.lang.String);
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: getstatic     #2                  // Field $assertionsDisabled:Z
       7: ifne          24
      10: aload_1
      11: ifnonnull     24
      14: new           #3                  // class java/lang/AssertionError
      17: dup
      18: ldc           #4                  // String Name must not be null!
      20: invokespecial #5                  // Method java/lang/AssertionError."<init>":(Ljava/lang/Object
      23: athrow
      24: aload_0
      25: aload_1
      26: putfield      #6                  // Field name:Ljava/lang/String;
      29: return

  public void setName(java.lang.String);
    Code:
```

```
    25: aload_1
    26: putfield      #6                  // Field name:Ljava/lang/String;
    29: return

public void setName(java.lang.String);
  Code:
     0: aload_0
     1: aload_1
     2: putfield      #6                  // Field name:Ljava/lang/String;
     5: return

public java.lang.String getName();
  Code:
     0: aload_0
     1: getfield      #6                  // Field name:Ljava/lang/String;
     4: areturn

public java.lang.String toString();
  Code:
     0: new           #7                  // class java/lang/StringBuilder
     3: dup
     4: invokespecial #8                  // Method java/lang/StringBuilder."<init>":()V
     7: ldc           #9                  // String Person(
     9: invokevirtual #10                 // Method java/lang/StringBuilder.append:(Ljava/lang/
                                          //                        String;)Ljava/lang/StringBuilder;
    12: aload_0
    13: getfield      #6                  // Field name:Ljava/lang/String;
    16: invokevirtual #10                 // Method java/lang/StringBuilder.append:(Ljava/lang/
                                          //                        String;)Ljava/lang/StringBuilder;
    19: ldc           #11                 // String )
    21: invokevirtual #10                 // Method java/lang/StringBuilder.append:(Ljava/lang/
                                          //                        String;)Ljava/lang/StringBuilder;
    24: invokevirtual #12                 // Method java/lang/StringBuilder.toString:()Ljava/lang/String
    27: areturn
```

```java
public class Person {
    public String name;
    public Person(String name) {
     assert name != null : "Name == null!";
     this.name = name;
    }
    public void setName(String name) {
     this.name = name;
    }
    public String getName() {
     return this.name;
    }
    public String toString() {
     return "Person(" + this.name + ")";
    }
}
```

Automatisk skräpsamling

- Mål: att ge programmeraren en illusion att minnet är oändligt
- Metod: identifiera *skräp-data* och frigör det **automatiskt**
- Definitionen av skräp: data som inte kan nås av programmet
- Mer formellt: objektet O är skräp om det inte finns någon väg i minnesgrafen från något rot (variabeln på stacken, globala variabler, o.dyl.) till O
- Två grundläggande sätt att göra automatisk skräpsamling:

  Referensräkning

  Tracing

Referensräkning

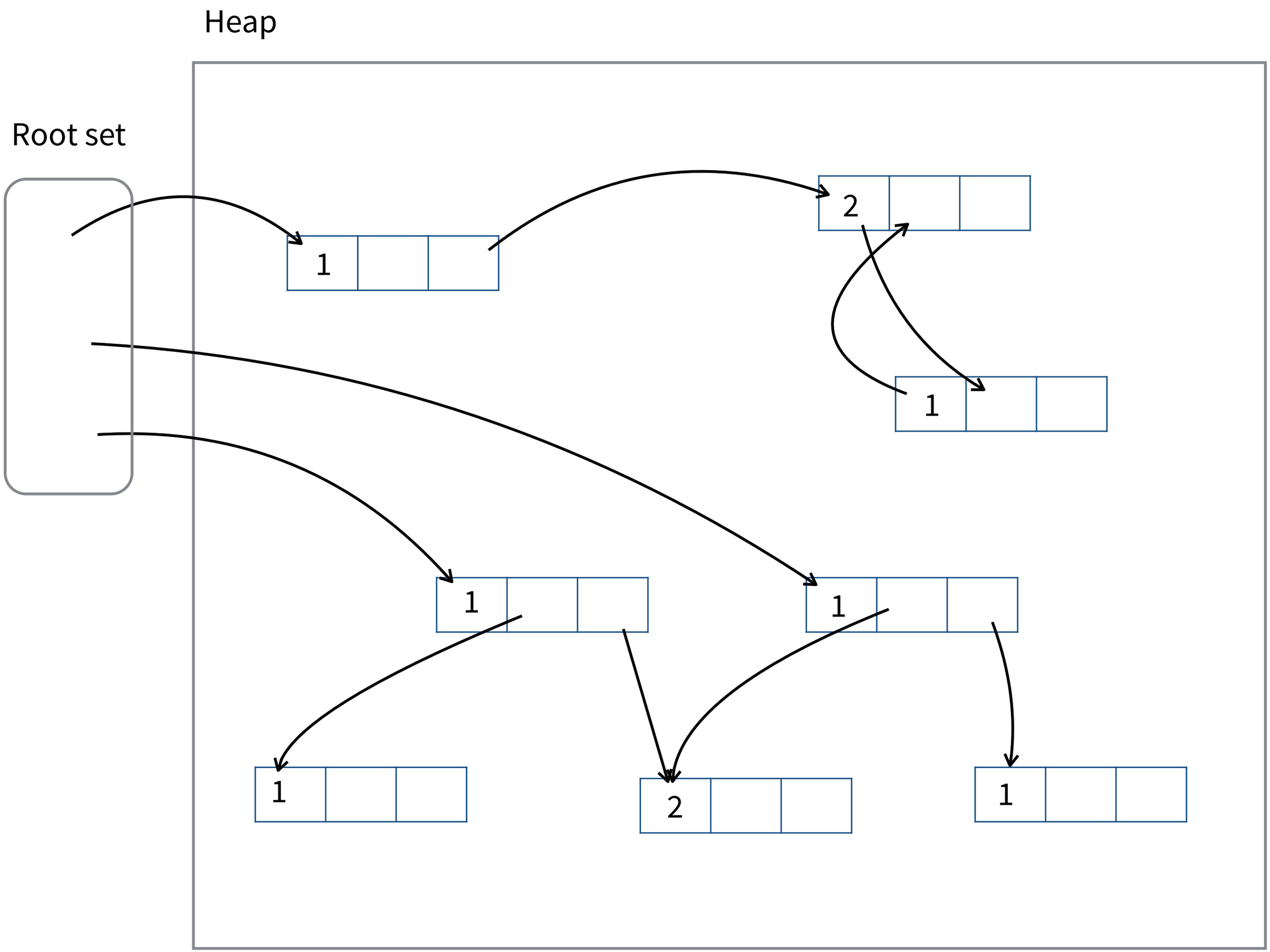
- Grundläggande idé: varje objekt sparar information om hur många som pekar till det
- När denna räknare når 0 — ta bort objektet
- Varje gång en referens skapas/tas bort, manipulera referensräknaren:

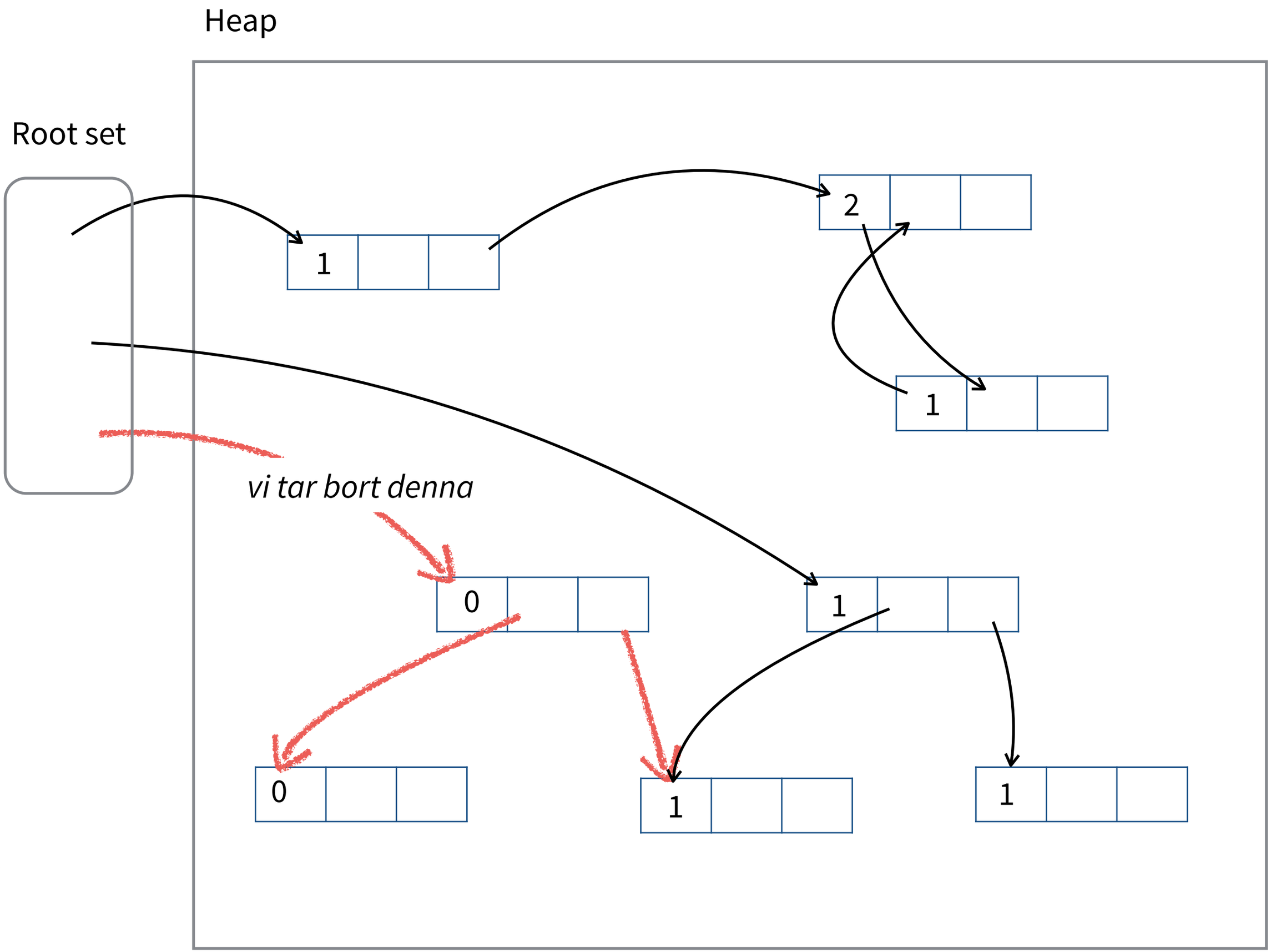
```
void *p = malloc(2048); // refcount 1
void *x = p; // refcount 2
p = NULL; // refcount 1
x = NULL; // refcount 0, free(x)
```

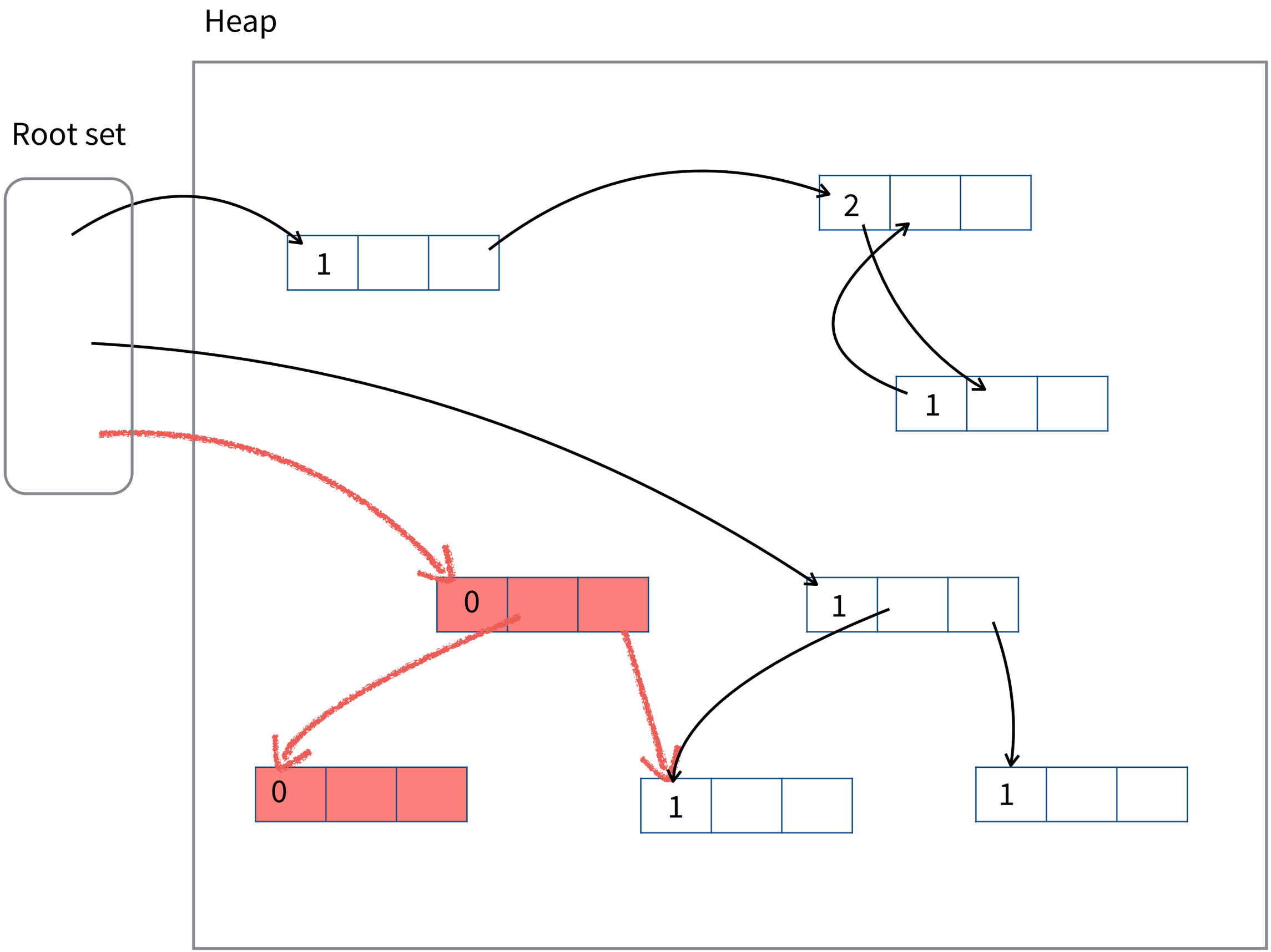
- Problem:

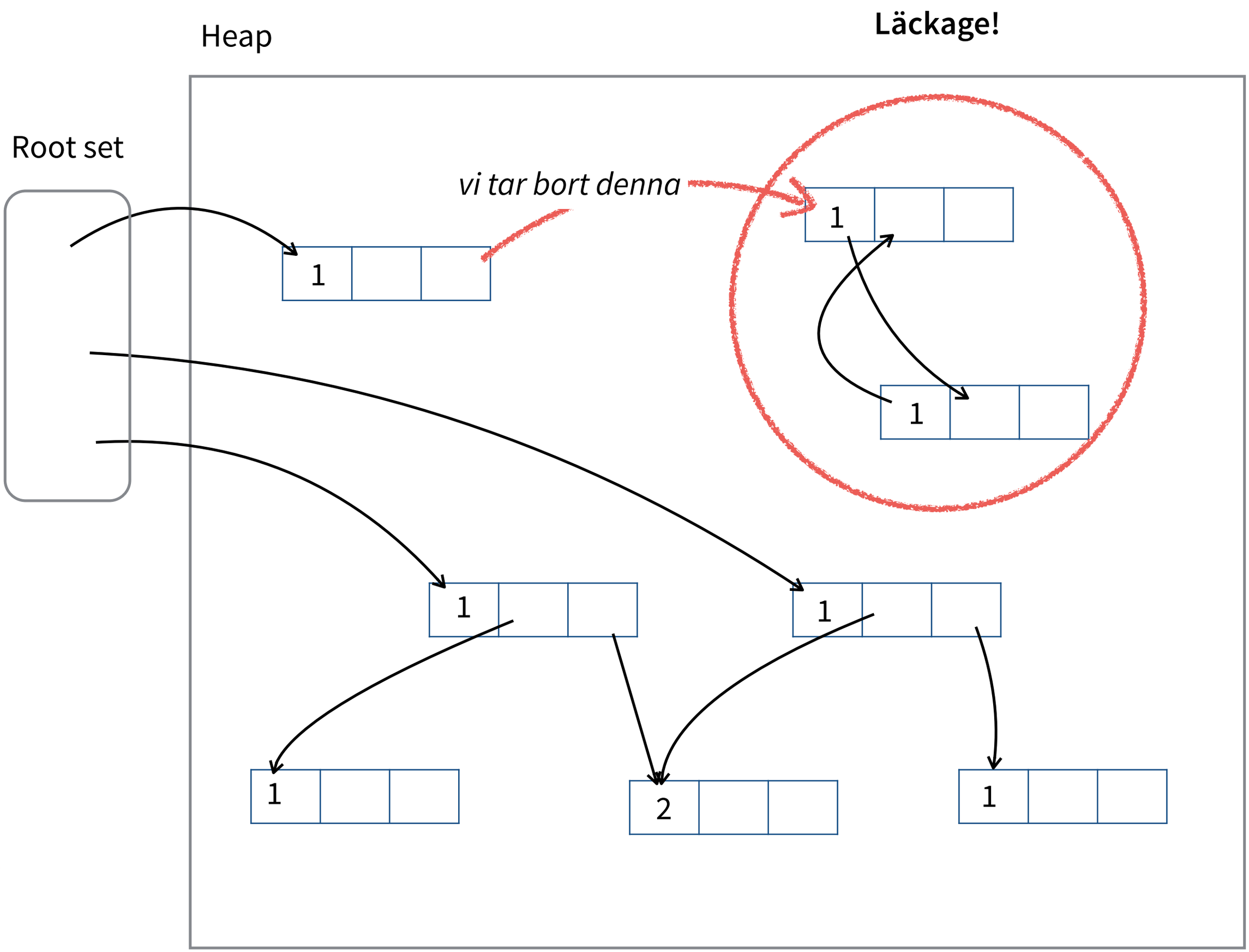
  Cykliska strukturer (se nästföljande sidor)

  Långlivat minne som manipuleras ofta kostar, fast vi aldrig tar bort det


Heap
Root set
1
2
1
1
1
1
2
1


Heap
Root set
1
2
1
vi tar bort denna
0
1
0
1
1


Heap
Root set
1
2
1
0
1
0
1
1

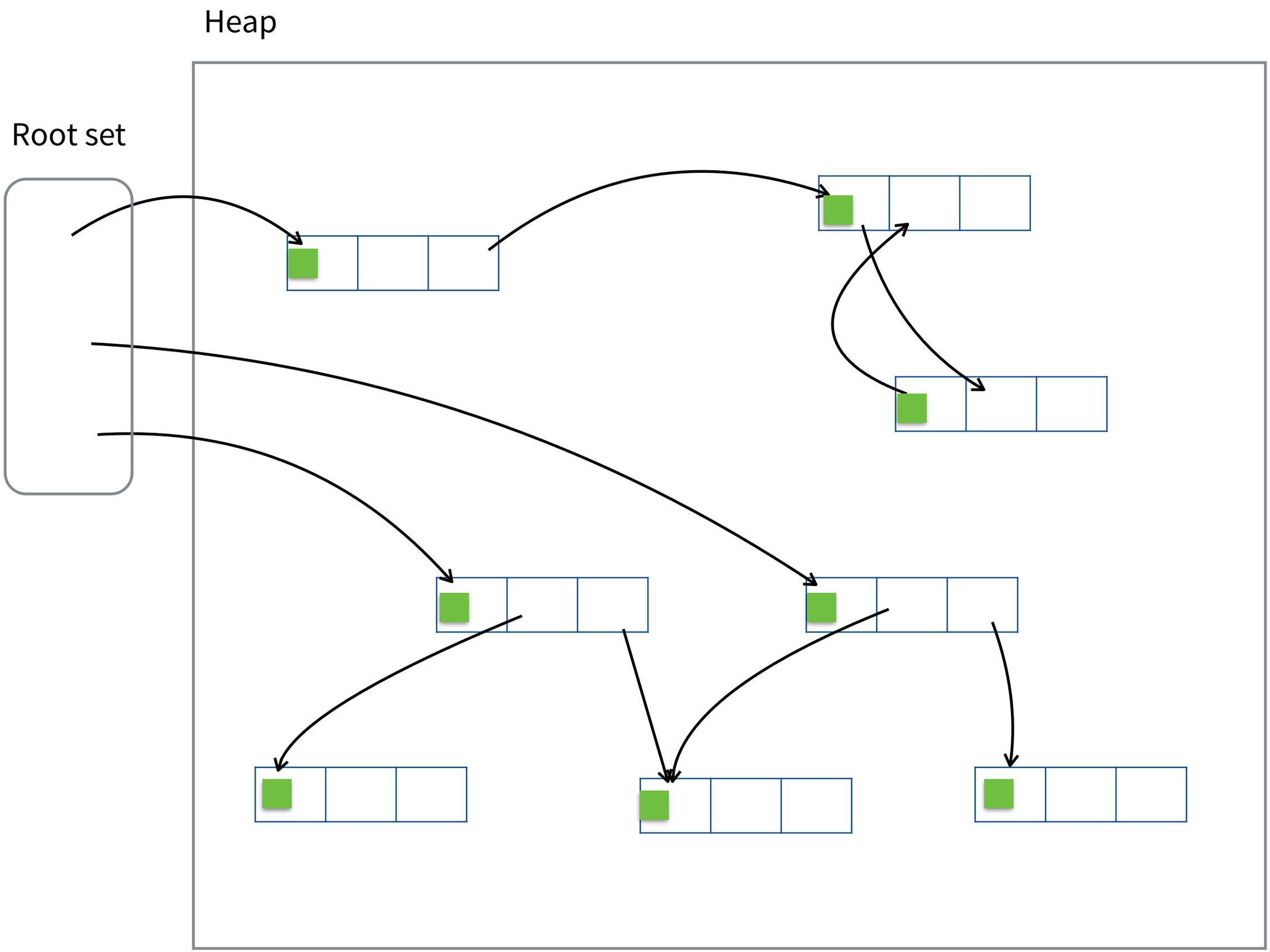

Läckage!
Heap
Root set
vi tar bort denna
1
1
1
1
1
1
2
1

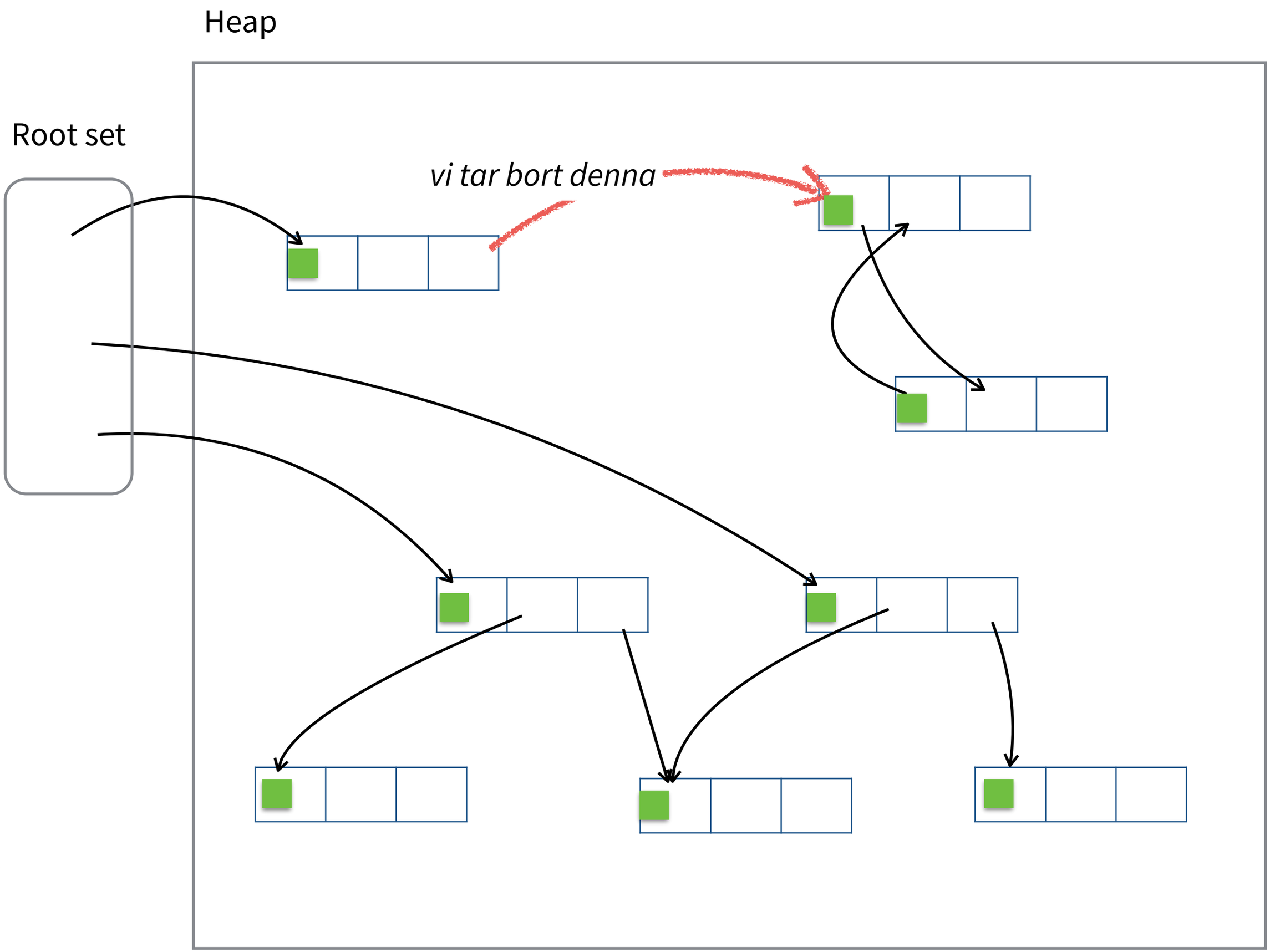
Tracing GC: Mark-Sweep

- När minnet tar slut:

  1. Följ rötterna och markera alla objekt som kan nås
  2. Iterera över alla objekt och frigör alla som inte markerats i 1.

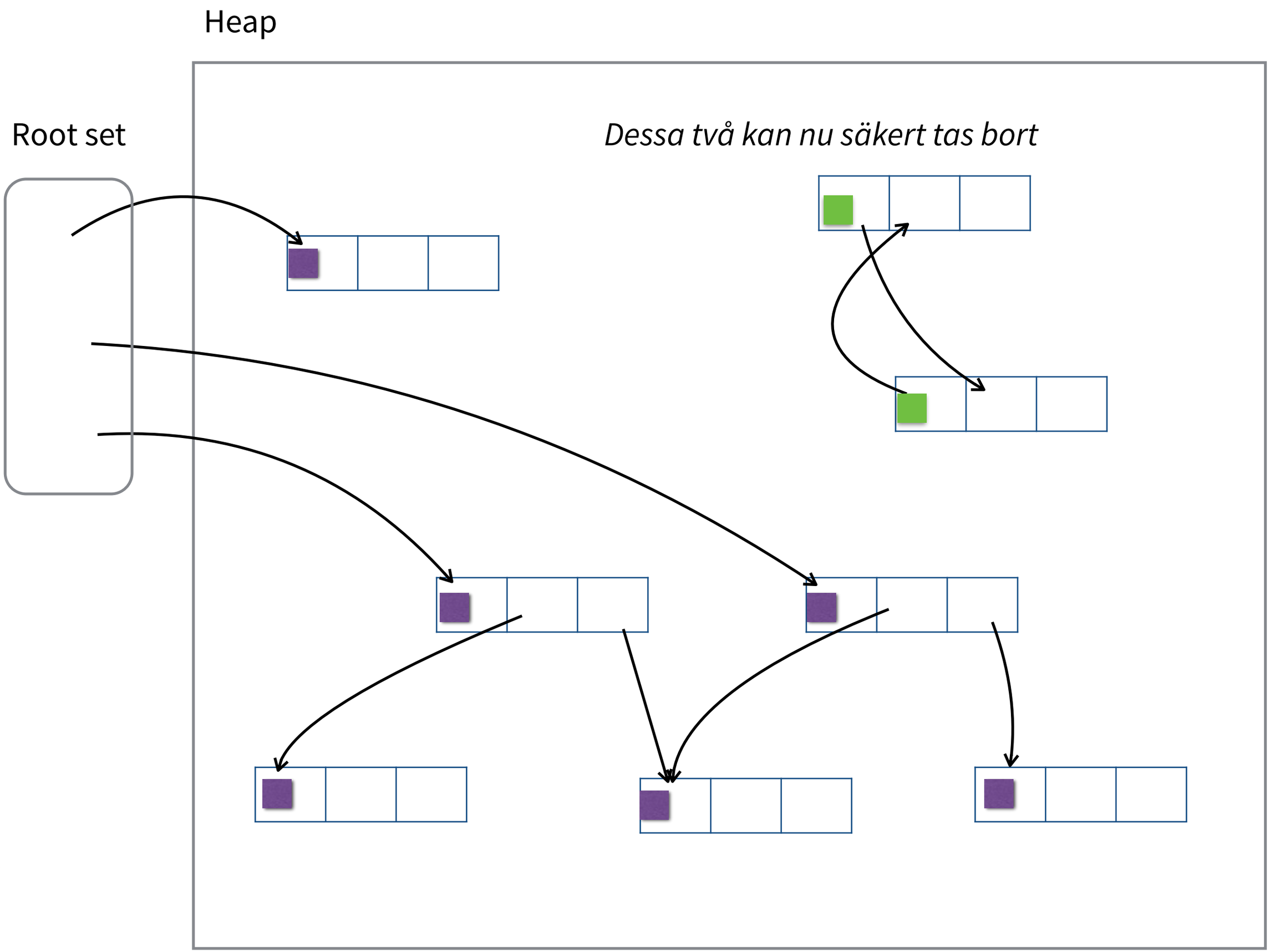
- Nästa bild visar markering i *mark-fasen (1.)*


Heap
Root set


Heap
Root set
vi tar bort denna

I nästa mark-fas markerar vi med annan färg

- Om något är grönt fortfarande efter denna fas är det skräp och skall tas bort


Heap
Root set
Dessa två kan nu säkert tas bort

KEEP
CALM
AND
LOVE
PROGRAMMING