Java Avancé

Memory

Emerite NEOU

Thread

Object Header

```
public class Point {
  private final int x ;
  private final y ;
}
```

Taille d'une instance une machine 64 bits ?

Object Header

```
public class Point {
  private final int x ;
  private final y ;
}
```

Taille d'une instance une machine 64 bits ?

- indice : taille d'un int 32 bits (4 octets)

Object Header

```
public class Point {
  private final int x ;
  private final y ;
}
```

Taille d'une instance une machine 64 bits ?

- indice : taille d'un int 32 bits (4 octets)
- reponse : 192 bits (24 octets)

Object Header

24 octets ?

Object Header

24 octets ?

- deux int : 8 octets

Object Header

24 octets ?

- deux int : 8 octets
- object header : 16 octets

Object Header - quezaco

object header ?

Object Header - quezaco

object header ?

- Mark word - 8 octets - synchornised/Object HashCode/GC
- Klass word - 8 octets - information sur la class de l'instance
- Size - 4 octets - taille du tableau (seulement pour un array)

Object Header - quezaco

```
synchronized(o) {...}
```

Object Header - quezaco

```
synchronized(o) {...}
```

```
o.myMethod() ;
```

Object Header - quezaco

```
synchronized(o) {...}
```

```
o.myMethod() ;
```

```
array.size() ;
```

int[] vs Integer[] vs ArrayList vs LinkedList

```
int[] array = new int[100] ;
```

- object header : 20 octets
- data size : 100*4 octets
- total : 420 octets

int[] vs Integer[] vs ArrayList vs LinkedList

```
int[] array = new int[100] ;
```

- object header : 20 octets
- data size : 100*4 octets
- total : 420 octets

```
Integer[] array = new Integer[100] ;
```

- object header : 20 octets
- Integer size : 20 octets (16 octets object header + 4 octets pour int)
- data size : 100*20 octets
- total : 2020 octets

int[] vs Integer[] vs ArrayList vs LinkedList

```
List<Integer> list = new ArrayList<>(100) ;
```

- object header : 16 octets
- data size : 100*4 octets
- total : 2016 octets

int[] vs Integer[] vs ArrayList vs LinkedList

```
List<Integer> list = new ArrayList<>(100) ;
```

- object header : 16 octets
- data size : 100*4 octets
- total : 2016 octets

```
List<Integer> list = new LinkedList<>(100) ;
```

- object header : 16 octets
- one cell size : 24 octets (pour une class Wrapper) + 20 octets (pour Integer)
- data size : 100*44 octets
- total : 4416 octets

Consequences

- GC

Consequences

- GC
- cache miss

Consequences

Table 2.2 Example Time Scale of System Latencies

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 µs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

Consequences

L1 cache : 256ko

- 12 integers : (
- 64 ints :)

Consequences

L1 cache : 256ko

- 12 integers : (
- 64 ints :)

parcours d'un tableau de 1000

- integers : 84 cache misses
- ints : 16 cache misses

Project Valhalla

- Value Types - permet de crée des objets sans object heade
- Generic Specialization - permet de crée des list de int