

Assignment 14

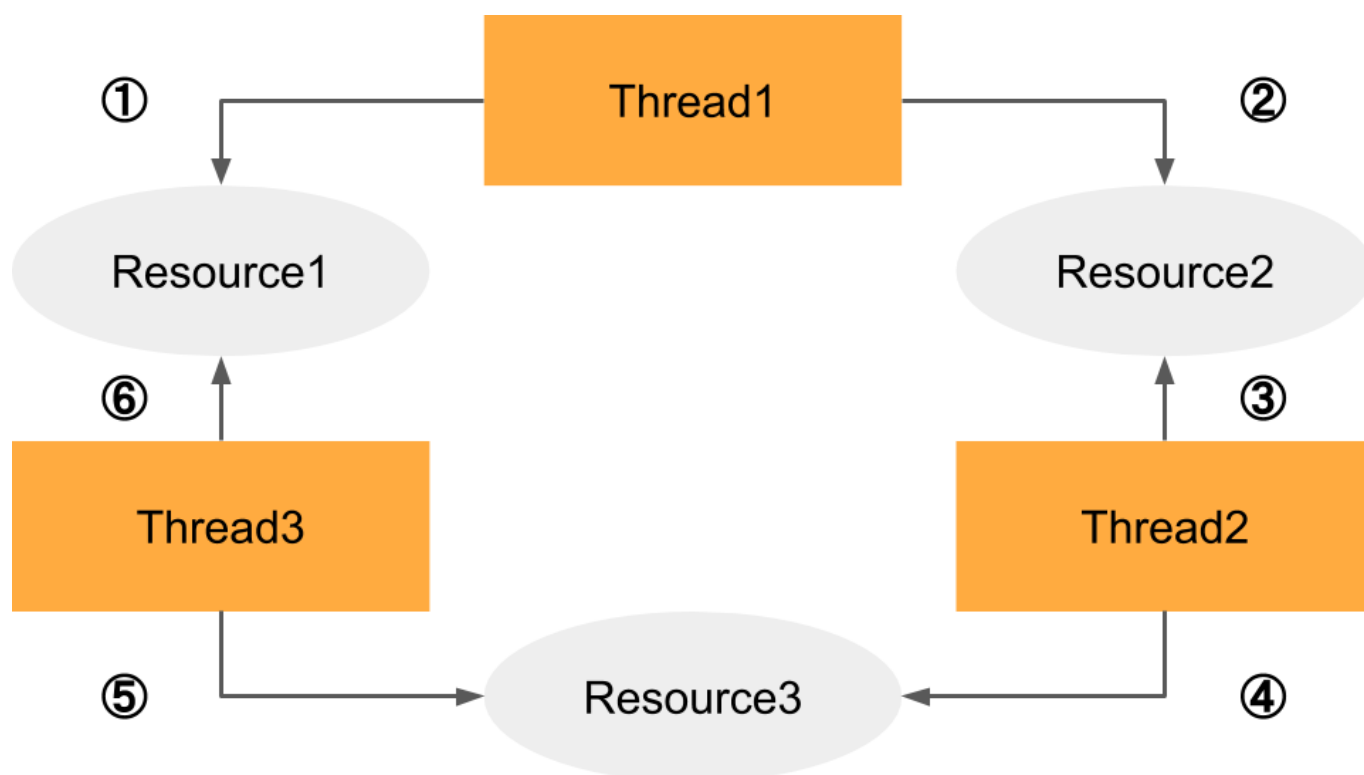
- 氏名: 栗林健太郎
- 学生番号: 2030006
- 作成日: 2020年12月4日

Deadlock

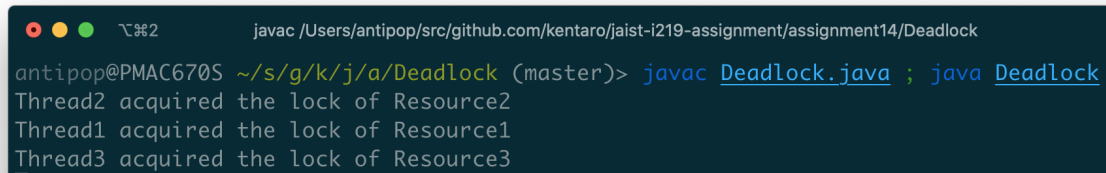
3つのスレッドと3つのリソースがある状態で、それぞれのスレッドは隣り合うリソースを確保して処理を行う
Deadlockクラスについて検討する。

Deadclass クラスの実行

Deadlock.javaに含まれているコードは、以下の図の通りリソースを循環的に参照しているため、デッドロックが発生する。



Deadlockクラスを実行すると、以下の通りデッドロックとなる。

A terminal window with a dark blue background. The title bar shows standard macOS window controls (red, yellow, green buttons) and the text "javac /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/Deadlock". The terminal content shows the user "antipop@PMAC670S" in the directory "~/s/g/k/j/a/Deadlock" (master) running the command "javac Deadlock.java ; java Deadlock". The output shows three threads acquiring locks: "Thread2 acquired the lock of Resource2", "Thread1 acquired the lock of Resource1", and "Thread3 acquired the lock of Resource3". A cursor is visible on the line following the last output.

```
javac /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/Deadlock  
antipop@PMAC670S ~/s/g/k/j/a/Deadlock (master)> javac Deadlock.java ; java Deadlock  
Thread2 acquired the lock of Resource2  
Thread1 acquired the lock of Resource1  
Thread3 acquired the lock of Resource3
```

Deadclassクラスの解析

JPFを実行すると、以下の通りデッドロックが検出された。

```

fish /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/Deadlock

owned locks:Resource@15f
blocked on: Resource@160
call stack:
    at Deadlock.run(Deadlock.java:21)

thread Deadlock:{id:3,name:Thread-3,status:BLOCKED,priority:5,isDaemon:false,lockCount:0,suspendCount:0}
    owned locks:Resource@160
    blocked on: Resource@15e
    call stack:
        at Deadlock.run(Deadlock.java:21)

===== results
error #1: gov.nasa.jpf.vm.NotDeadlockedProperty "deadlock encountered: thread java.lang.Thread:{i..."

===== statistics
elapsed time:      00:00:00
states:           new=837,visited=1105,backtracked=1925,end=4
search:           maxDepth=44,constraints=0
choice generators: thread=836 (signal=0,lock=203,sharedRef=509,threadApi=33,reschedule=91),data=0
heap:             new=1719,released=1239,maxLive=384,gcCycles=1771
instructions:     18297
max memory:       245MB
loaded code:      classes=63,methods=1478

===== search finished: 20/12/04 23:55
antipop@PMAC670S ~/s/g/k/j/a/Deadlock (master)>

```

以下に、デッドロックに関するtransitionを抜粋する。

```

----- transition #6
thread: 1
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,1/3,isCascaded:false}
    Deadlock.java:19          : synchronized(resource1) {
    Deadlock.java:20          : System.out.println(name + " acquired
the lock of " + resource1.getName());
        [2 insn w/o sources]
    Deadlock.java:20          : System.out.println(name + " acquired
the lock of " + resource1.getName());
----- transition #10
thread: 1
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/3,isCascaded:false}
    Deadlock.java:21          : synchronized(resource2) {
----- transition #11
thread: 2
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,2/3,isCascaded:false}
    [1 insn w/o sources]
    Deadlock.java:1           : /**

```

```

    Deadlock.java:19          : synchronized(resource1) {
----- transition #17
thread: 2
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/2,isCascaded:false}
    Deadlock.java:21          : synchronized(resource2) {
----- transition #18
thread: 3
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,2/2,isCascaded:false}
    [1 insn w/o sources]
    Deadlock.java:1          : /**
    Deadlock.java:19          : synchronized(resource1) {
----- transition #20
thread: 3
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,2/2,isCascaded:false}
    Deadlock.java:21          : synchronized(resource2) {
    .run(Deadlock.java:21)

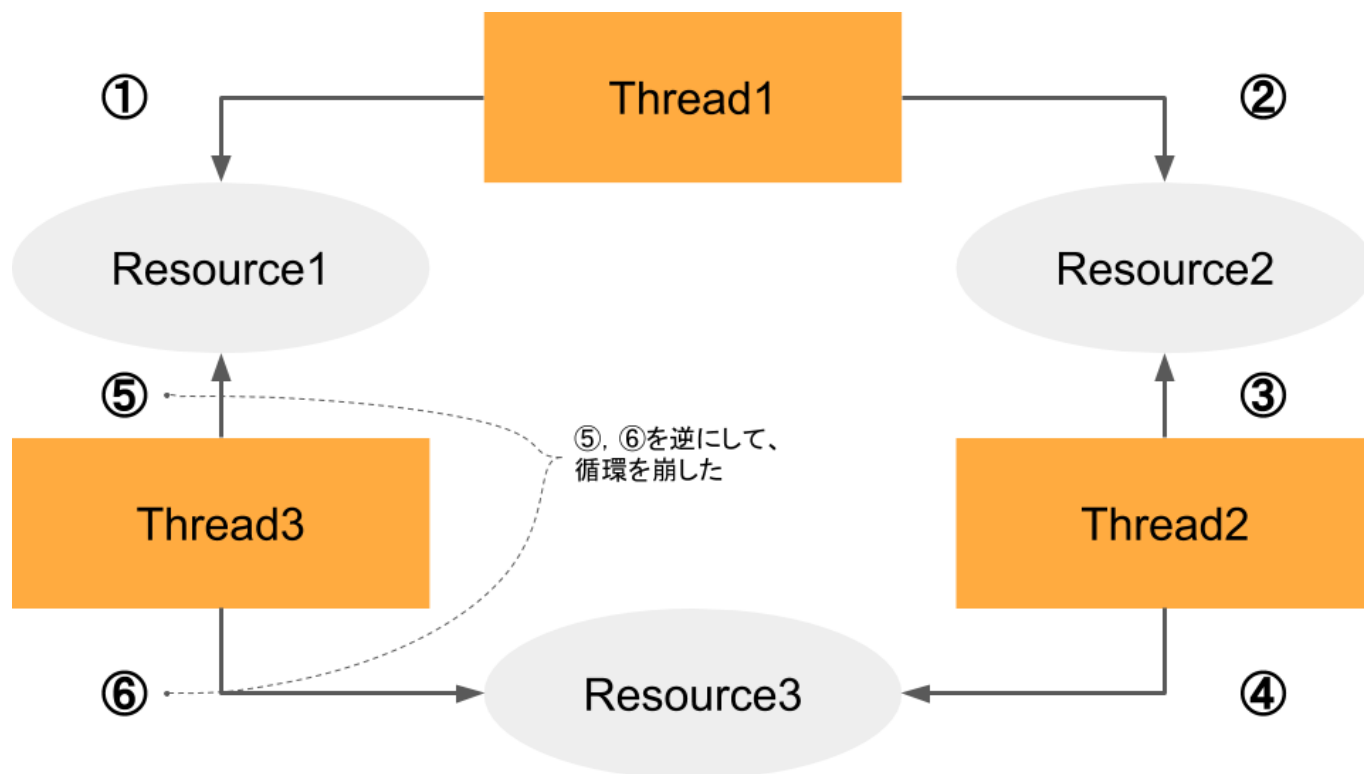
```

上記のtransitionの内容は以下の通りである。

1. Thread1がResource1のロックを確保
2. Thread2がResource2のロックを確保
3. Thread3がResource3のロックを確保
4. Thread1がResource2のロックを待つ
5. Thread2がResource3のロックを待つ
6. Thread3がResource1のロックを待つ

このため、**Deadlock**クラスの実行によってデッドロックが発生する。

ちなみに、下図のように、たとえばThread3のロック確保順を逆にして循環を崩すと、デッドロックせずにプログラムが終了する。



実行結果は以下の通りである。

```

fish /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/Deadlock
antipop@PMAC670S ~/s/g/k/j/a/Deadlock (master)> javac Deadlock.java ; java Deadlock
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread1 acquired the lock of Resource1
Thread1 acquired the lock of Resource1
Thread3 acquired the lock of Resource1
Thread3 acquired the lock of Resource1
Resource1: 2, Resource2: 2, Resource3: 2
antipop@PMAC670S ~/s/g/k/j/a/Deadlock (master)>

```

JPFを実行しても、デッドロックは検出されない。

```

fish /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/Deadlock

Thread1 acquired the lock of Resource1
Thread1 acquired the lock of Resource1
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread2 acquired the lock of Resource2
Thread1 acquired the lock of Resource1
Thread1 acquired the lock of Resource1

===== results
no errors detected

===== statistics
elapsed time:      00:00:01
states:           new=5075,visited=9480,backtracked=14555,end=4
search:           maxDepth=58,constraints=0
choice generators: thread=5075 (signal=0,lock=1016,sharedRef=3302,threadApi=51,reschedule=706), data=0
heap:             new=7977,released=7127,maxLive=388,gcCycles=13168
instructions:     98789
max memory:       309MB
loaded code:      classes=63,methods=1479

===== search finished: 20/12/05 0:27
antipop@PMAC670S ~/s/g/k/j/a/Deadlock (master)>

```

DiningPhilosopherProblem

DiningPhilosopherProblemを下記のように変更することで、デッドロックが起ることを確認した。

```

// changed the argument `n - 1` to `n`
// to confirm it if causes a deadlock.
DiningRoom dr = new DiningRoom(n);

```

JPFで検査すると、以下の通りデッドロックが検出された。

```

fish /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/DPPDeadlock

waiting on: Chopstick@166
call stack:
  at java.lang.Object.wait(Object.java)
  at Chopstick.acquire(Chopstick.java:16)
  at Philosopher.run(Philosopher.java:33)

===== results
error #1: gov.nasa.jpf.vm.NotDeadlockedProperty "deadlock encountered:  thread
Philosopher:{id:1,n..."

===== statistics
elapsed time:      00:00:00
states:           new=4542,visited=6744,backtracked=11256,end=62
search:           maxDepth=56,constraints=0
choice generators: thread=4541 (signal=321,lock=1585,sharedRef=1950,threadApi=3
, reschedule=682), data=0
heap:             new=392,released=6466,maxLive=381,gcCycles=8343
instructions:      74806
max memory:       309MB
loaded code:      classes=66,methods=1484

===== search finished: 20/12/08
20:29
antipop@PMAC670S ~/s/g/k/j/a/DPPDeadlock (master)>

```

以下に、デッドロックに関するtransitionを抜粋する。

```

----- transition #8
thread: 1
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,1/3,isCascaded:false}
  Philosopher.java:27      : droom.enter();
  DiningRoom.java:17       : if (howManyPeopleCanEnter > cnt) {
  DiningRoom.java:18       : cnt++;
  DiningRoom.java:24       : }
  Philosopher.java:28      : } catch (InterruptedException e) {}
  Philosopher.java:30      : left.acquire();

----- transition #9
thread: 1
gov.nasa.jpf.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,1/3,isCascaded:false}
  Philosopher.java:30      : left.acquire();
  Chopstick.java:15        : while (beingUsed) {
  Chopstick.java:18        : beingUsed = true;
  Chopstick.java:19        : }
  Philosopher.java:31      : } catch (InterruptedException e) {}
  Philosopher.java:33      : right.acquire();

----- transition #14
thread: 2

```

```

gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,2/3,isCascaded:false}
  DiningRoom.java:18          : cnt++;
  DiningRoom.java:24          : }
  Philosopher.java:28         : } catch (InterruptedException e) {}
  Philosopher.java:30         : left.acquire();
----- transition #15
thread: 2
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,2/3,isCascaded:false}
  Philosopher.java:30         : left.acquire();
  Chopstick.java:15          : while (beingUsed) {
----- transition #16
thread: 2
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/2,isCascaded:false}
  Chopstick.java:15          : while (beingUsed) {
  Chopstick.java:18          : beingUsed = true;
----- transition #17
thread: 2
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/2,isCascaded:false}
  Chopstick.java:18          : beingUsed = true;
  Chopstick.java:19          : }
----- transition #19
thread: 1
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/3,isCascaded:false}
  Chopstick.java:15          : while (beingUsed) {
  Chopstick.java:16          : this.wait();
  [1 insn w/o sources]
----- transition #20
thread: 2
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"WAIT"
,1/2,isCascaded:false}
  Chopstick.java:19          : }
  Philosopher.java:31         : } catch (InterruptedException e) {}
  Philosopher.java:33         : right.acquire();
----- transition #26
thread: 3
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"LOCK"
,2/2,isCascaded:false}
  Philosopher.java:30         : left.acquire();
  Chopstick.java:15          : while (beingUsed) {
  Chopstick.java:18          : beingUsed = true;
  Chopstick.java:19          : }
----- transition #27
thread: 2
gov.nasa.jpff.vm.choice.ThreadChoiceFromSet {id:"RELEASE"
,1/2,isCascaded:false}
  Philosopher.java:33         : right.acquire();
  Chopstick.java:15          : while (beingUsed) {
----- transition #28
thread: 2

```



```

gov.nasa.jpjf.vm.choice.ThreadChoiceFromSet {id:"SHARED_OBJECT"
,1/2,isCascaded:false}
  Chopstick.java:15          : while (beingUsed) {
  Chopstick.java:16          : this.wait();
    [1 insn w/o sources]
----- transition #29
thread: 3
gov.nasa.jpjf.vm.choice.ThreadChoiceFromSet {id:"WAIT"
,1/1,isCascaded:false}
  Chopstick.java:19          : }
  Philosopher.java:31       : } catch (InterruptedException e) {}
  Philosopher.java:33       : right.acquire();
  Chopstick.java:15          : while (beingUsed) {
  Chopstick.java:16          : this.wait();
    [1 insn w/o sources]

```

上記のtransitionの内容は以下の通りである。

1. **thread: 1**が、**transition #8**で左の箸を取り、**transition: 9**で右の箸を取るを試みる
2. **thread: 2**が、**transiton #14-17**で左の箸を取る
3. そのため、**thread: 1**は**transition #18-19**で右の箸を取れるまで待つ
4. **thread: 2**が、**transition #20**で右の箸の箸を取るを試みる
5. **thread: 3**が、**transition #26**左の箸を取る
6. そのため、**thread: 2**は**transition #28**で右の箸を取れるまで待つ
7. 最後に、**thread: 3**は**transition #29**で右の箸を取れるまで待つ

このため、**DiningPhilosopherProblem**クラスの実行によってデッドロックが発生する。

NewSolutionToDPP

DiningPhilosopherProblemに対する別解を考える。**DiningPhilosopherProblem**は、ダイニングルームに入ることでできる人数を制限することで、デッドロックを防ぐアプローチであった。そのため、前述の通り、哲学者がn人いる場合に、n-1人ではなくn人全員がダイニングルームに入ることができるようにすると、デッドロックを引き起こす状況となった。

別解として、ダイニングルームへの人数制限を行わないアプローチについて考える。その場合、**Deadlock**のところで見た通り、リソースへの参照が循環性を持たないようにすることが必要である。そのため、奇数番目の哲学者は左・右の順番で、偶数番目の哲学者は右・左の順番で箸を取るようにすることで、循環性を崩すようにした（実装の変更の主要な部分は以下の通り）。

また、循環性を壊すだけなら哲学者の内のひとりが右・左という順番で箸をとるだけでいいが、奇数・偶数にわけることによって並行性をできるだけ損うことなく、デッドロックも回避できる。

```

7c7
< public class DiningPhilosopherProblem {
---
> public class NewSolutionToDPP {
11d10
<         DiningRoom dr = new DiningRoom(n);
16c15,16

```

```

<      for (int i = 0; i < (n - 1); i++) {
----
>      int i = 0;
>      for (i = 0; i < (n - 1); i++) {
19c19,23
<          (new Philosopher(m, left, right, dr)).start();
----
>          if (i % 2 == 0) {
>              (new Philosopher(m, left, right)).start();
>          } else {
>              (new Philosopher(m, right, left)).start();
>          }
23c27,32
<      (new Philosopher(m, left, right, dr)).start();
----
>
>      if (i % 2 == 0) {
>          (new Philosopher(m, left, right)).start();
>      } else {
>          (new Philosopher(m, right, left)).start();
>      }

```

上記の別解をJPFで検査する。

まず、哲学者が3人の場合で検査し、以下の通りエラーなく終了したことを確認した。

```

fish /Users/antipop/src/github.com/kentaro/jaist-1219-assignment/assignment14/NewSolutionToDPP
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master)> grep "dpp.begin(3, 1)" TestNewSolutionToDPP.java
    dpp.begin(3, 1);
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master)> javac *.java; time jpf TestNewSolutionToDPP.jpf
JavaPathfinder core system v8.0 (rev 31) - (C) 2005-2014 United States Government. All rights reserved.

===== system under test
TestNewSolutionToDPP.main()

===== search started: 20/12/08 22:12

===== results
no errors detected

===== statistics
elapsed time:      00:00:01
states:            new=8760,visited=17143,backtracked=25903,end=64
search:            maxDepth=62,constraints=0
choice generators: thread=8760 (signal=959,lock=2100,sharedRef=3449,threadApi=3,reschedule=2249), data=0
heap:              new=2794,released=35731,maxLive=377,gcCycles=20703
instructions:      204534
max memory:        434MB
loaded code:       classes=65,methods=1481

===== search finished: 20/12/08 22:12

-----
Executed in   1.96 secs   fish           external
  usr time    5.37 secs   83.00 micros   5.37 secs
  sys time    0.26 secs   405.00 micros   0.26 secs
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master)>

```

次に、哲学者が5人の場合でも検査し、以下の通りエラーなく終了したことを確認した。

```

fish /Users/antipop/src/github.com/kentaro/jaist-i219-assignment/assignment14/NewSolutionToDPP
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master) [1]> grep "dpp.begin(5, 1)" TestNewSolutionToDPP.java
    dpp.begin(5, 1);
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master)> javac *.java; time jpf TestNewSolutionToDPP.jpf
JavaPathfinder core system v8.0 (rev 31) - (C) 2005-2014 United States Government. All rights reserved.

===== system under test
TestNewSolutionToDPP.main()

===== search started: 20/12/08 22:15

===== results
no errors detected

===== statistics
elapsed time:      00:05:22
states:           new=2052678,visited=7349879,backtracked=9402557,end=106
search:           maxDepth=104,constraints=0
choice generators: thread=2052678 (signal=220128,lock=494653,sharedRef=813526,threadApi=5,reschedule=524366), data=0
heap:             new=630770,released=9900745,maxLive=387,gcCycles=7282773
instructions:     64637860
max memory:       502MB
loaded code:      classes=65,methods=1481

===== search finished: 20/12/08 22:20

-----
Executed in 323.07 secs  fish           external
   usr time 332.74 secs  100.00 micros  332.74 secs
   sys time  2.51 secs   546.00 micros    2.51 secs
antipop@PMAC670S ~/s/g/k/j/a/NewSolutionToDPP (master)>

```