

```

beta.java          1/39
~/tmp/           02/17/2011

import java.util.Random;
import java.util.*;
import java.util.*;

class Foo {
    public static void main(String[] args) {
        System.out.println("aaarg");
    }
}

public class Beta {
    static Random rand = new Random();
    private static final double gam(int x) {
        double result = 0;
        for (int i = 1; i <= x; i++) {
            result += Math.log(rand.nextDouble());
        }
        return result;
    }
}

class Thing {
    public Thing() {
        Runtime.getRuntime().addShutdownHook(new Thread() {
            public void run() { flush(); }
        });
    }
    public void flush() { /* do some deferred action */ }
    public static void main(String[] args) {
        Thing t;
        while (true) { t = new Thing(); }
    }
}

class Crash {
    public static void main(String[] args) {
        Object[] o = new Object[1];
        while (null != o[0] = new Object());
    }
}

class GenericDemo {
    public static <T> Iterator<T> collapse(final Iterator<? extends T> source
        ) {
        return new Iterator<T>() {
            private Iterator<? extends T> buffer = null;
            public void remove() {
                throw new UnsupportedOperationException("Not supported");
            }
            public T next() {
                if (buffer == null) {
                    if (hasNext())
                        throw new NoSuchElementException("No more elements");
                }
                return buffer.next();
            }
        };
    }
}

```

beta.java
~ /tmp/

2/39
02/17/2011
beta.java
~/.tmp/

2/39
02/17/2011
beta.java
~/.tmp/

beta.java
~/tmp/
3/39
02/17/2011

```
144     while (null != o) { o = new Object[]{o}; }
145   }
146 }
147
148 class GenericDemo {
149   public static <T> Iterator<T> collapse(final Iterator<? extends T> source
150   ) {
151     return new Iterator<T>() {
152       private Iterator<? extends T> buffer = null;
153
154       public void remove() {
155         throw new UnsupportedOperationException("Not supported");
156       }
157
158       public T next() {
159         if (!hasNext()) {
160           throw new NoSuchElementException("No more elements");
161         }
162         return buffer.next();
163       }
164
165       public boolean hasNext() {
166         while (source.hasNext()) {
167           if (null == buffer)
168             || !buffer.hasNext()) {
169               buffer = source.next();
170             }
171         }
172         return buffer.hasNext();
173       }
174     };
175   }
176
177   public static void main(String[] args) {
178     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
179     List<String> l = new ArrayList<String>();
180     l.add("foo");
181     l.add("bar");
182     li.add(l.iterator());
183
184     l = new ArrayList<String>();
185     l.add("baz");
186     l.add("qux");
187     li.add(l.iterator());
188
189     Iterator<String> i = collapse(li.iterator());
190     while (!i.hasNext()) {
191       System.out.println(i.next());
192     }
193   }
194
195   public static void foo(aarg) {
196     System.out.println("aaarg");
197   }
198
199   public class beta {
200     static Random rand = new Random();
201
202     private static final double gamin(int x) {
203       double result = 0;
204       for (int i = 1; i <= x; i++) {
205         result += Math.log(rand.nextDouble());
206       }
207     }
208
209     public static double drawint(a, int b) {
210       return gamma() / (gamin(a) + gamin(b));
211     }
212
213   }
214
215   public static void main(String[] args) {
```

beta.java
~/tmp/
4/39
02/17/2011

```
145     for (int i = 0; i < 100000; i++) {
146       System.out.println(draw(2, 5));
147     }
148   }
149
150   class Thing {
151     public Thing() {
152       Runtime.getRuntime().addShutdownHook(new Thread() {
153         public void run() {
154           flush();
155         }
156       });
157     }
158
159     public void flush() {
160       /* do some deferred action */
161       Thing t = new Thing();
162       while (true) {
163         t.new Thing();
164       }
165     }
166
167     public static void main(String[] args) {
168       Object[] o = new Object[1];
169       o[0] = new Object();
170       while (null != o) {
171         o[0] = new Object();
172       }
173     }
174
175     class crash {
176       public static void main(String[] args) {
177         Object[] o = new Object[1];
178         o[0] = new Object();
179         while (null != o) {
180           o[0] = new Object();
181         }
182       }
183
184       class GenericDemo {
185         public static <T> Iterator<T> collapse(final Iterator<? extends T> source
186         ) {
187           return new Iterator<T>() {
188             private Iterator<? extends T> buffer = null;
189
190             public void remove() {
191               throw new UnsupportedOperationException("Not supported");
192             }
193
194             public T next() {
195               if (!hasNext()) {
196                 throw new NoSuchElementException("No more elements");
197               }
198             }
199
200             public boolean hasNext() {
201               public void hasNext() {
202                 public T next() {
203                   if (!hasNext())
204                     throw new NoSuchElementException("No more elements");
205                   return buffer.next();
206                 }
207               }
208             }
209
210             public void add(T o) {
211               buffer.add(o);
212             }
213
214             public void remove() {
215               buffer.remove();
216             }
217
218             public void close() {
219               buffer.close();
220             }
221
222             public void flush() {
223               Runtime.getRuntime().addShutdownHook(new Thread() {
224                 public void run() {
225                   flush();
226                 }
227               });
228             }
229
230             public static void main(String[] args) {
231               Thing t = new Thing();
232               while (true) {
233                 t.new Thing();
234               }
235             }
236           }
237         }
238
239         class GenericDemo {
240           public static <T> Iterator<T> collapse(final Iterator<? extends T> source
241           ) {
242             return new Iterator<T>() {
243               private Iterator<? extends T> buffer = null;
244
245               public void remove() {
246                 throw new UnsupportedOperationException("Not supported");
247               }
248
249               public void hasNext() {
250                 throw new UnsupportedOperationException("Not supported");
251               }
252
253               public T next() {
254                 if (!hasNext())
255                   throw new NoSuchElementException("No more elements");
256                 return buffer.next();
257               }
258             }
259           }
260
261           public void add(T o) {
262             buffer.add(o);
263             if (null == buffer)
264               buffer = source.next();
265             else
266               buffer = source.next();
267             }
268           }
269
270           public static void main(String[] args) {
271             List<Iterator<String>> li = new ArrayList<Iterator<String>>();
272             List<String> l = new ArrayList<String>();
273             l.add("foo");
274             l.add("bar");
275             li.add(l.iterator());
276             l = new ArrayList<String>();
277             l.add("baz");
278             li.add(l.iterator());
279             Iterator<String> i = collapse(li.iterator());
280             while (!i.hasNext()) {
281               System.out.println(i.next());
282             }
283           }
284
285         }
286       }
287     }
288
289   }
290
291   public static void main(String[] args) {
292     System.out.println("aaarg");
293   }
294 }
```

beta.java
~/tmp/
5/39
02/17/2011

```
266 }
267     || ibuffer.hasNext() ) {
268         buffer = source.next();
269     }
270     public static void aaarg() { System.out.println("aaarg"); }
271 }
272 
273 public class beta {
274     static Random rand = new Random();
275     private static final double gamma( int x ) {
276         double result = 0;
277         for ( int i = 1; i <= x; i++ ) {
278             result += Math.log(rand.nextDouble());
279         }
280     }
281     return result;
282 }
283 
284 public static double draw( int a, int b ) {
285     return gamma( a ) / ( gamma(a) + gamma(b) );
286 }
287 
288 public static void main( String[] args ) {
289     for ( int i = 0; i < 100000; i++ ) {
290         System.out.println(draw(2, 5));
291     }
292 }
293 
294 class Thing {
295     public Thing() {
296         Runtime.getRuntime().addShutdownHook( new Thread() {
297             public void run() { flush(); }
298         });
299     }
300     public void flush() { /* do some deferred action */ }
301     public static void main( String[] args ) {
302         Thing t;
303         while ( true ) { t = new Thing(); }
304     }
305 }
306 
307 class GenericDemo {
308     public static < T > Iterator< T > collapse( final Iterator< ? extends T > source
309     ) {
310         return new Iterator< T >() {
311             private Iterator< ? extends T > buffer = null;
312             public void remove() {
313                 throw new UnsupportedOperationException( "Not supported" );
314             }
315             public T next() {
316                 if ( buffer.hasNext() ) {
317                     return buffer.next();
318                 }
319                 if ( null == o ) {
320                     o = new Object();
321                 }
322                 public static void main( String[] args ) {
323                     Object[] o = new Object[1];
324                     while ( null != o ) { o = new Object[1][o]; }
325                 }
326             }
327         }
328     }
329 }
330 
331 class Crash {
332     public static void main( String[] args ) {
333         Object[] o = new Object[1];
334         while ( null != o ) { o = new Object[1][o]; }
335     }
336 }
337 
338 class GenericDemo {
339     public static < T > Iterator< T > collapse( final Iterator< ? extends T > source
340     ) {
341         return new Iterator< T >() {
342             private Iterator< ? extends T > buffer = null;
343             public void remove() {
344                 throw new UnsupportedOperationException( "Not supported" );
345             }
346             public T next() {
347                 if ( buffer.hasNext() ) {
348                     return buffer.next();
349                 }
350                 if ( null == o ) {
351                     o = new Object();
352                 }
353                 public boolean hasNext() {
354                     while ( source.hasNext() ) {
355                         Object[] o = new Object[1];
356                         if ( null == buffer ) {
357                             buffer = source.next();
358                         }
359                         o[0] = buffer;
360                         buffer = o;
361                         return true;
362                     }
363                 }
364             }
365         }
366     }
367 }
```

beta.java
~/tmp/
6/39
02/17/2011

```
357         buffer = source.next();
358     }
359     return buffer.hasNext();
360 }
361 }
362 }
363 }
364 }
365 }
366 public static void main( String[] args ) {
367     List< Iterator< String > > li = new ArrayList< Iterator< String > >();
368     List< String > l = new ArrayList< String >();
369     l.add( "foo" );
370     l.add( "bar" );
371     li.add( l.iterator() );
372     li.add( bar.iterator() );
373     li.add( bar.iterator().next() );
374     li.add( bar.iterator().next() );
375     Iterator< String > i = collapse( li.iterator() );
376     while ( i.hasNext() ) {
377         while ( System.out.print( i.next() ) );
378     }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 public class Beta {
389     static Random rand = new Random();
390     private static final double gamma( int x ) {
391         double result = 0;
392         for ( int i = 1; i <= x; i++ ) {
393             result += Math.log( rand.nextDouble() );
394         }
395     }
396     public static void main( String[] args ) {
397         for ( int i = 0; i < 100000; i++ ) {
398             System.out.println( draw( 2, 5 ) );
399         }
400     }
401 }
402 
403 class Foo {
404     public void flush() {
405         System.out.println( draw( 2, 5 ) );
406     }
407 }
408 
409 class Thing {
410     public Thing() {
411         Runtime.getRuntime().addShutdownHook( new Thread() {
412             public void run() {
413                 flush();
414             }
415         });
416     }
417     public void flush() {
418         System.out.println( draw( 2, 5 ) );
419     }
420     public static void main( String[] args ) {
421         Thing t;
422         while ( true ) { t = new Thing(); }
423     }
424 }
425 
426 class Crash {
427     public static void main( String[] args ) {
428         Object[] o = new Object[1];
429         if ( null == o ) {
430             throw new NoSuchElementException( "No more elements" );
431         }
432         return buffer.next();
433     }
434 }
435 
```

beta.java 7/39 02/17/2011

```

429     while (null != o) { o = new Object[]{o}; }
430   }
431 }
432 class GenericDemo {
433     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
434     ) {
435         return new Iterator<T>() {
436             private Iterator<? extends T> buffer = null;
437             public void remove() {
438                 throw new UnsupportedOperationException("Not supported");
439             }
440             public T next() {
441                 if (!hasNext()) {
442                     throw new NoSuchElementException("No more elements");
443                 }
444                 return buffer.next();
445             }
446             public boolean hasNext() {
447                 while (source.hasNext()) {
448                     if (null == buffer)
449                         buffer = source.next();
450                     else
451                         buffer = buffer.hasNext() ?
452                             null : buffer;
453                 }
454             }
455             return buffer.hasNext();
456         };
457     }
458 }
459 }
460 public static void main(String[] args) {
461     List<Iterator<String> li = new ArrayList<Iterator<String>>();
462     List<String> l = new ArrayList<String>();
463     l.add("foo");
464     l.add("bar");
465     li.add(l.iterator());
466     l = new ArrayList<String>();
467     l.add("baz");
468     l.add("qux");
469     li.add(l.iterator());
470     Iterator<String> i = collapse(li.iterator());
471     while (!i.hasNext()) {
472         System.out.println(i.next());
473     }
474 }
475 }
476 }
477 }
478 class foo {
479     public static void aaarg() { System.out.println("aaarg"); }
480 }
481 }
482 public class beta {
483     static Random rand = new Random();
484     static final double gam(int x) {
485         double result = 0;
486         for (int i = 1; i <= x; i++) {
487             result += Math.log(rand.nextDouble());
488         }
489     }
490     return result;
491 }
492 }
493 }
494 }
495 class foo {
496     public static double drawint( a, int b) {
497         return gamma / (gam(a) + gam(b));
498     }
499 }
500 }
501 public static void main(String[] args) {
502 }
```

beta.java 8/39 02/17/2011

```

429     for (int i = 0; i < 100000; i++) {
430         System.out.println(draw(2, 5));
431     }
432 }
433 class Thing {
434     public void run() {
435         Runtime.getRuntime().addShutdownHook(new Thread() {
436             public void run() { flush(); }
437         });
438     }
439     public void flush() { /* do some deferred action */ }
440     public static void main(String[] args) {
441         Thing t = new Thing();
442         t.run();
443     }
444 }
445 class crash {
446     public static void main(String[] args) {
447         Object[] o = new Object[1];
448         o[0] = new Object();
449         while (true) { o[0] = new Object(); }
450     }
451 }
452 class GenericDemo {
453     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
454     ) {
455         return new Iterator<T>() {
456             private Iterator<? extends T> buffer = null;
457             public void remove() {
458                 throw new UnsupportedOperationException("Not supported");
459             }
460             public T next() {
461                 if (!hasNext())
462                     throw new NoSuchElementException("No more elements");
463                 public boolean hasNext() {
464                     while (null != buffer)
465                         buffer = buffer.hasNext();
466                 }
467                 return buffer.hasNext();
468             }
469             public boolean hasNext() {
470                 if (null == buffer)
471                     throw new NoSuchElementException("No more elements");
472                 return buffer.next();
473             }
474             public void remove() {
475                 throw new UnsupportedOperationException("Not supported");
476             }
477         };
478     }
479 }
480 }
481 }
482 public static void main(String[] args) {
483     List<Iterator<String> li = new ArrayList<Iterator<String>>();
484     List<String> l = new ArrayList<String>();
485     l.add("foo");
486     l.add("bar");
487     li.add(l.iterator());
488     Iterator<String> i = collapse(li.iterator());
489     while (!i.hasNext()) {
490         System.out.println(i.next());
491     }
492 }
493 }
494 }
495 class foo {
496     public static void aaarg() { System.out.println("aaarg"); }
497 }
498 }
499 }
500 }
501 public static void main(String[] args) {
502 }
```

beta.java
~/tmp/
9/39
02/17/2011

```
571 }
572
573 class foo {
574     public static void aaarg() { System.out.println("aaarg"); }
575 }
576 }
577
578 public class beta {
579     static Random rand = new Random();
580
581     private static final double gamma1 = 0;
582     private static final double gamma2 = 1;
583
584     public static double draw(int a, int b) {
585         double result = 0;
586         for (int i = 1; i <= x; i++) {
587             result += Math.log(rand.nextDouble());
588         }
589         return result;
590     }
591
592     public static void main(String[] args) {
593         for (int i = 0; i < 100000; i++) {
594             System.out.println(draw(2, 5));
595         }
596     }
597
598 }
599 }
600
601 class Thing {
602     public Thing() {
603         Runtime.getRuntime().addShutdownHook(new Thread() {
604             public void run() { flush(); }
605         });
606     }
607
608     public void flush() { /* do some deferred action */ }
609
610     public static void main(String[] args) {
611         Thing t;
612         while (true) { t = new Thing(); }
613     }
614 }
615
616 class crash {
617     public static void main(String[] args) {
618         Object[] o = new Object[1];
619         while (null != o) { o = new Object[1][o]; }
620     }
621 }
622
623 class genericDemo {
624     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
625     ) {
626         return new Iterator<T>() {
627             private Iterator<? extends T> buffer = null;
628
629             public void remove() {
630                 throw new UnsupportedOperationException("Not supported");
631             }
632
633             public boolean hasNext() {
634                 if (hasNext()) (NoSuchElementException) {
635                     throw new NoSuchElementException("No more elements");
636                 }
637                 return buffer.hasNext();
638             }
639
640             public T next() {
641                 if (hasNext()) {
642                     if (buffer.hasNext()) {
643                         return buffer.next();
644                     }
645                     while (source.hasNext()) {
646                         if (null == buffer
647                             && (null = buffer
```

beta.java
~/tmp/
10/39
02/17/2011

```
642             || !buffer.hasNext()) {
643                 buffer = source.next();
644             }
645             return buffer.hasNext();
646         };
647     };
648 }
649
650     public static void main(String[] args) {
651         ArrayList<String> li = new ArrayList<String>();
652         List<String> l = new ArrayList<String>();
653         l.add("foo");
654         l.add("bar");
655         li.addAll(l.iterator());
656         li.add(li.iterator());
657         li = new ArrayList<String>();
658         li.addAll(li.add("qux"));
659         li.add(li.iterator());
660         Iterator<String> i = collapse(li.iterator());
661         while (!i.hasNext()) {
662             System.out.println(i.next());
663         }
664     }
665
666     class foo {
667         public static void aaarg() {
668             System.out.println("aaarg");
669         }
670     }
671
672     public class beta {
673         static Random rand = new Random();
674
675         private static final double gamma1 = x;
676         double result = 0;
677         for (int i = 1; i <= x; i++) {
678             result += Math.log(rand.nextDouble());
679         }
680         return result;
681     }
682
683         public static double draw(int a, int b) {
684             return gamma1 / (gamma(a) + gamma(b));
685         }
686
687         public static void main(String[] args) {
688             for (int i = 0; i < 100000; i++) {
689                 System.out.println(draw(2, 5));
690             }
691         }
692
693         class Thing {
694             public Thing() {
695                 Runtime.getRuntime().addShutdownHook(new Thread() {
696                     public void run() { flush(); }
697                 });
698             }
699         }
700
701         public void flush() { /* do some deferred action */ }
702
703         public static void main(String[] args) {
704             Thing t;
705             t = new Thing();
706             while (true) { t = new Thing(); }
707         }
708
709         class crash {
710             public static void main(String[] args) {
711                 Object[] o = new Object[1];
712                 o = new Object[1];
713                 while (null != o) { o = new Object[1][o]; }
714             }
715         }
716     }
717 }
```

beta.java
~/tmp/
11/39
02/17/2011

```
74      }
75  }
76  class GenericDemo {
77    public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
78  ) {
79    return new Iterator<T>() {
80      private Iterator<? extends T> buffer = null;
81      public void remove() {
82        throw new UnsupportedOperationException("Not supported");
83      }
84      public T next() {
85        if (!hasNext()) {
86          throw new NoSuchElementException("No more elements");
87        }
88        return buffer.next();
89      }
90      public boolean hasNext() {
91        while (source.hasNext()) {
92          if (null == buffer)
93            buffer = source.next();
94          if (buffer.hasNext())
95            return buffer.hasNext();
96        }
97      }
98      public static void main(String[] args) {
99        List<String> li = new ArrayList<String>();
100       li.add("foo");
101       li.add("bar");
102       li.add(li.iterator());
103       li = new ArrayList<String>();
104       li.add("baz");
105       li.add("qux");
106       li.add(li.iterator());
107       Iterator<String> i = collapse(li.iterator());
108       while (!i.hasNext()) {
109         System.out.println(i.next());
110       }
111     }
112   }
113   class foo {
114     public static void aaarg() {
115       System.out.println("aaarg");
116     }
117   }
118   public class beta {
119     static Random rand = new Random();
120     private static final double gamma() {
121       double result = 0;
122       for (int i = 1; i < x; i++) {
123         result += Math.log(rand.nextDouble());
124       }
125     }
126     return result;
127   }
128   static void main(String[] args) {
129     System.out.println(draw(2, 5));
130   }
131 }
132 
```

beta.java
~/tmp/
12/39
02/17/2011

```
786      System.out.println(draw(2, 5));
787    }
788  }
789  class Thing {
790    public Thing() {
791      Runtime.getRuntime().addShutdownHook(new Thread() {
792        public void run() {
793          public void flush() {
794            /* do some deferred action */
795          }
796        }
797        public static void main(String[] args) {
798          public static void main(String[] args) {
799            public void main(String[] args) {
800              Thing t;
801              while (true) { t = new Thing(); }
802            }
803          }
804        }
805        class crash {
806          public static void main(String[] args) {
807            Object[] o = new Object[1];
808            while (null != o) { o = new Object[0]; }
809          }
810        }
811        class GenericDemo {
812          public static Iterator<?> collapse(final Iterator<? extends Iterator<? extends T>> source
813          ) {
814            return new Iterator<Iterator<?>>() {
815              private Iterator<?> buffer = null;
816              public void remove() {
817                throw new UnsupportedOperationException("Not supported");
818              }
819              public T next() {
820                if (!hasNext())
821                  throw new NoSuchElementException("No more elements");
822                return buffer.next();
823              }
824              public void remove() {
825                throw new UnsupportedOperationException("Not supported");
826              }
827              public boolean hasNext() {
828                while (source.hasNext())
829                  if (hasNext())
830                    if (buffer == buffer
831                      || !buffer.hasNext())
832                      buffer = source.next();
833                }
834              return buffer.next();
835            };
836          }
837        }
838        public static void main(String[] args) {
839          List<String> li = new ArrayList<String>();
840          li.add("foo");
841          li.add("bar");
842          li.add(li.iterator());
843          li = new ArrayList<String>();
844          li.add("baz");
845          li.add("qux");
846          li.add(li.iterator());
847          Iterator<String> i = collapse(li.iterator());
848          while (!i.hasNext()) {
849            System.out.println(i.next());
850          }
851        }
852      }
853    }
854  }
855 }
```

beta.java
~/tmp/
13/39
02/17/2011

```
836
837     class foo {
838         public static void main(String[] args) {
839             System.out.println("aaarg");
840         }
841     }
842     public class beta {
843         static Random rand = new Random();
844         static final double gamma = 0;
845         public static double draw(int a, int b) {
846             double result = 0;
847             for (int i = 1; i <= b; i++) {
848                 result += Math.log(rand.nextDouble());
849             }
850             return result;
851         }
852     }
853     public static void main(String[] args) {
854         for (int i = 0; i < 100000; i++) {
855             System.out.println(draw(2, 5));
856         }
857     }
858     class Thing {
859         public Thing() {
860             Runtime.getRuntime().addShutdownHook(new Thread() {
861                 public void run() { flush(); }
862             });
863         }
864         public void flush() { /* do some deferred action */ }
865         public static void main(String[] args) {
866             Thing t;
867             while (true) { t = new Thing(); }
868         }
869     }
870     class GenericDemo {
871         public static <T> Iterator<T> collapse(final Iterator<? extends T> source
872             ) {
873             return new Iterator<T>() {
874                 private Iterator<? extends T> buffer = null;
875                 public void remove() {
876                     throw new UnsupportedOperationException("Not supported");
877                 }
878             };
879         }
880     }
881     class crash {
882         public static void main(String[] args) {
883             Object[] o = new Object[1];
884             o[0] = new Object[1][o];
885             while (null != o) { o = new Object[1][o]; }
886         }
887     }
888     class GenericDemo {
889         public static <T> Iterator<T> collapse(final Iterator<? extends T> source
890             ) {
891                 return new Iterator<T>() {
892                     private Iterator<? extends T> buffer = null;
893                     public void remove() {
894                         throw new UnsupportedOperationException("Not supported");
895                     }
896                 };
897             }
898         }
899     }
900     class crash {
901         public static void main(String[] args) {
902             Object[] o = new Object[1];
903             o[0] = new Object[1][o];
904             while (null != o) { o = new Object[1][o]; }
905         }
906     }
907     class crash {
908         public static void main(String[] args) {
909             Object[] o = new Object[1];
910             o[0] = new Object[1][o];
911             public void remove() {
912                 throw new UnsupportedOperationException("Not supported");
913             }
914         }
915         public T next() {
916             if (hasNext()) {
917                 if (hasNext()) {
918                     throw new NoSuchElementException("No more elements");
919                 }
920                 return buffer.next();
921             }
922         }
923         public boolean hasNext() {
924             while (source.hasNext() && (null == buffer ||
925                 || !buffer.hasNext())) {
926             }
927         }
928         buffer = source.next();
929         return buffer.hasNext();
930     }
931     }
932     class crash {
933         public static void main(String[] args) {
934             Object[] o = new Object[1];
935             while (null != o) { o = new Object[1][o]; }
936         }
937     }
938 }
```

beta.java
~/tmp/
14/39
02/17/2011

```
927         buffer = source.next();
928     }
929     return buffer.hasNext();
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
999 }
```

```

15/3/9
02/17/2011

beta.java
~/tmp/
```

```

    }
}
}

class GenericDemo<T> Iterator<T> collapse(final Iterator<? extends T>> source
{
    public static <T> Iterator<T> new Iterator<?>() {
        return new Iterator<?>() {
            private Iterator<?> extends T> buffer = null;
            public void remove() {
                throw new UnsupportedOperationException("Not supported");
            }
            public T next() {
                if (buffer.hasNext()) {
                    throw new NoSuchElementException("No more elements");
                }
                return buffer.next();
            }
            public boolean hasNext() {
                while (source.hasNext())
                    && (null == buffer
                        || buffer.hasNext());
                buffer = source.next();
            }
            return buffer.hasNext();
        };
    }
}

public static void main(String[] args) {
    List<String> listString = new ArrayList<Iterator<String>>();
    listString.add(new ArrayList<String>());
    listString.add(new ArrayList<String>());
    listString.add(new ArrayList<String>());
    listString.add(new ArrayList<String>());
    Iterator<String> iterator = collapse(listString.iterator());
    while (iterator.hasNext()) {
        System.out.println(iterator.next());
    }
}

class foo {
    static void aarg() {
        System.out.println("aarg");
    }
}

public static final double gam(int x) {
    double result = 0;
    for (int i = 1; i <= x; i++) {
        result += Math.log(i).nextDouble();
    }
    return result;
}

public static double draw(int a, int b) {
    return gam(a) / (gam(a) + gam(b));
}

public static void main(String[] args) {
    for (int i = 0; i < 10000; i++) {
        public static double draw(int a, int b) {
            return gam(a) / (gam(a) + gam(b));
        }
        public static void main(String[] args) {
            for (int i = 0; i < 10000; i++) {

```

beta.java
~ /tmp/

18/39
02/17/2011
beta.java
~/tmp/

```

class foo {
    public static void aaargs() { System.out.println("aaargs"); }
}

public class beta {
    static Random rand = new Random();
    private static final double gamma = 1.0;
    private static final double gama = 1.0;
    public static double draw(int a, int b) {
        return gamma / (gama(a) + gama(b));
    }
    public static void main(String[] args) {
        for (int i = 1; i <= x; i++) {
            result += Math.log(rand.nextDouble());
        }
        return result;
    }
}

class Thing {
    public Thing() {
        Runtime.getRuntime().addShutdownHook(new Thread() {
            public void run() { flush(); }
        });
    }
    public void flush() { /* do some deferred action */ }
    public static void main(String[] args) {
        Thing t;
        while (true) { t = new Thing(); }
    }
}

class crash {
    public static void main(String[] args) {
        Object[] o = new Object[1];
        while (null != o) { o = new Object[o]; }
    }
}

class GenericDemo {
    public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
    ) {
        return new Iterator<T>() {
            private Iterator<? extends T> buffer = null;
            public void remove() {
                throw new UnsupportedOperationException("Not supported");
            }
            public T next() {
                if (buffer == null) {
                    throw new NoSuchElementException("No more elements");
                }
                return buffer.next();
            }
            public boolean hasNext() {
                while (source.hasNext()) {
                    if (hasNext()) {
                        throw new NoSuchElementException("No more elements");
                    }
                    buffer = buffer.next();
                }
            }
        };
    }
}

```

19/39
02/17/2011
beta.java
~/tmp/

beta.java
~/tmp/

20/39
02/17/2011
beta.java
~\tmp\

20/39
02/17/2011
beta.java
~\tmp\

```

}
}

1234
1235
1236 class GenericDemo {  

1237     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source  

1238     ) {  

1239         return new Iterator<T>() {
1240             private Iterator<?> buffer = null;
1241
1242             public void remove() {
1243                 throw new UnsupportedOperationException("Not supported");
1244             }
1245
1246             public T next() {
1247                 if (buffer == null) {
1248                     throw new NoSuchElementException("No more elements");
1249                 }
1250                 return buffer.next();
1251             }
1252
1253             public boolean hasNext() {
1254                 while (source.hasNext())
1255                     if (null == buffer
1256                         || !buffer.hasNext())
1257                         buffer = source.next();
1258                 return buffer.hasNext();
1259             }
1260
1261             public void add(T element) {
1262                 throw new UnsupportedOperationException("Not supported");
1263             }
1264
1265         };
1266     }
1267
1268     public static void main(String[] args) {
1269         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1270         List<String> l1 = new ArrayList<String>();
1271         l1.add("foo");
1272         li.add(l1.iterator());
1273
1274         l1 = new ArrayList<String>();
1275         l1.add("bar");
1276         li.add(l1.iterator());
1277
1278         Iterator<String> i = collapse(li.iterator());
1279         while (i.hasNext()) {
1280             System.out.println(i.next());
1281         }
1282
1283         class foo {
1284             public static void main(String[] args) {
1285                 System.out.println("aarg");
1286             }
1287         }
1288
1289         static Random rand = new Random();
1290
1291         public class beta {
1292             private static final double gamma = 0.37;
1293             static Random rand = new Random();
1294
1295             public static double draw(int a, int b) {
1296                 double result = 0;
1297                 for (int i = 1; i <= x; i++) {
1298                     result += Math.log(rand.nextDouble());
1299                 }
1300                 return result;
1301             }
1302
1303             public static double draw(int a, int b) {
1304                 return gamma / (gamma * a + gamma * b);
1305             }
1306
1307             public static void main(String[] args) {
1308                 for (int i = 0; i < 1000000; i++) {
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
343
```

beta.java
~/tmp/
21/39
02/17/2011

```
1426
1427     class foo {
1428         public static void main(String[] args) {
1429             System.out.println("aaarg");
1430         }
1431     }
1432     public class beta {
1433         static Random rand = new Random();
1434         private static final double gammaInt = 0;
1435         double result = 0;
1436         for (int i = 1; i <= x; i++) {
1437             result += Math.log(rand.nextDouble());
1438         }
1439         return result;
1440     }
1441     public static double draw(int a, int b) {
1442         return (gamma(a) / (gamma(a) + gamma(b)));
1443     }
1444     public static void main(String[] args) {
1445         for (int i = 0; i < 1000000; i++) {
1446             System.out.println(draw(2, 5));
1447         }
1448     }
1449     class Thing() {
1450         public Runtime getRuntime() {
1451             return Runtime.getRuntime();
1452         }
1453     }
1454     public static void main(String[] args) {
1455         Thing t;
1456         while (true) { t = new Thing(); }
1457     }
1458     public void run() { flush(); }
1459     }
1460     public void flush() { /* do some deferred action */ }
1461     public static void main(String[] args) {
1462         public static void main(String[] args) {
1463             public static void main(String[] args) {
1464                 Thing t;
1465                 while (true) { t = new Thing(); }
1466             }
1467         }
1468     }
1469     class GenericDemo {
1470         public static <T> Iterator<T> collapse(final Iterator<? extends T> source
1471             ) {
1472             return new Iterator<T>() {
1473                 private Iterator<? extends T> buffer = null;
1474             }
1475         }
1476         class GenericDemo {
1477             public static <T> Iterator<T> collapse(Iterator<? extends T> source
1478                 ) {
1479                     throw new UnsupportedOperationException("Not supported");
1480                 }
1481             }
1482         }
1483         public void remove() {
1484             throw new UnsupportedOperationException();
1485         }
1486         public T next() {
1487             if (!hasNext()) {
1488                 throw new NoSuchElementException("No more elements");
1489             }
1490             return buffer.next();
1491         }
1492         public boolean hasNext() {
1493             while (source.hasNext()
1494                 && (null == buffer
1495                     || !buffer.hasNext())) {
1496
1497             }
1498             buffer = source.next();
1499             return buffer.hasNext();
1500         }
1501     }
1502     public static void main(String[] args) {
1503         List<String> li = new ArrayList<String>();
1504         li.add("foo");
1505         li.add("bar");
1506         li.add("bar");
1507         li.iterator();
1508         li = new ArrayList<String>();
1509         li.add("bar");
1510         li.add("qux");
1511         li.iterator();
1512         li.add("bar");
1513         li.add("qux");
1514         Iterator<String> i = collapse(li.iterator());
1515         while (i.hasNext()) {
1516             System.out.println(i.next());
1517         }
1518     }
1519     public static void foo() {
1520         System.out.println("aaarg");
1521     }
1522     public class beta {
1523         static Random rand = new Random();
1524         private static final double gammaInt = 0;
1525         double result = 0;
1526         for (int i = 1; i <= x; i++) {
1527             result += Math.log(rand.nextDouble());
1528         }
1529         return result;
1530     }
1531     public static double draw(int a, int b) {
1532         double result = 0;
1533         for (int i = 1; i <= x; i++) {
1534             result += Math.log(rand.nextDouble());
1535         }
1536         return result;
1537     }
1538     public static double draw(int a, int b) {
1539         public static final double gammaInt = 0;
1540         return gamma(a) / (gamma(a) + gamma(b));
1541     }
1542     public static void main(String[] args) {
1543         for (int i = 0; i < 100000; i++) {
1544             System.out.println(draw(2, 5));
1545         }
1546     }
1547     class Thing {
1548         public Thing() {
1549             Runtime.getRuntime().addShutdownHook(new Thread() {
1550                 public void run() {
1551                     public Thing() {
1552                         Runtime.getRuntime().addShutdownHook(new Thread() {
1553                             public void run() {
1554                             }
1555                         });
1556                     }
1557                 }
1558             });
1559         }
1560         public void flush() {
1561             /* do some deferred action */
1562         }
1563         public static void main(String[] args) {
1564             public static void main(String[] args) {
1565                 Thing t;
1566                 while (true) { t = new Thing(); }
1567             }
1568         }
1569         class crash {
1570             public static void main(String[] args) {
1571                 Object[] o = new Object[1];
1572                 Object[] o = new Object[1];
1573                 while (null != o) { o = new Object[1]; }
1574             }
1575         }
1576         public static void main(String[] args) {
1577             public static void main(String[] args) {
1578                 Object[] o = new Object[1];
1579                 Object[] o = new Object[1];
1580                 while (null != o) { o = new Object[1]; }
1581             }
1582         }
1583     }
1584 }
```

beta.java
~/tmp/
22/39
02/17/2011

```
1426
1427     class foo {
1428         public static void main(String[] args) {
1429             System.out.println("aaarg");
1430         }
1431     }
1432     public class beta {
1433         static Random rand = new Random();
1434         private static final double gammaInt = 0;
1435         double result = 0;
1436         for (int i = 1; i <= x; i++) {
1437             result += Math.log(rand.nextDouble());
1438         }
1439         return result;
1440     }
1441     public static double draw(int a, int b) {
1442         return (gamma(a) / (gamma(a) + gamma(b)));
1443     }
1444     public static void main(String[] args) {
1445         for (int i = 0; i < 1000000; i++) {
1446             System.out.println(draw(2, 5));
1447         }
1448     }
1449     class Thing() {
1450         public Runtime getRuntime() {
1451             return Runtime.getRuntime();
1452         }
1453     }
1454     public static void main(String[] args) {
1455         Thing t;
1456         while (true) { t = new Thing(); }
1457     }
1458     public void run() { flush(); }
1459     }
1460     public void flush() { /* do some deferred action */ }
1461     public static void main(String[] args) {
1462         public static void main(String[] args) {
1463             public static void main(String[] args) {
1464                 Thing t;
1465                 while (true) { t = new Thing(); }
1466             }
1467         }
1468     }
1469     class GenericDemo {
1470         public static <T> Iterator<T> collapse(final Iterator<? extends T> source
1471             ) {
1472             return new Iterator<T>() {
1473                 private Iterator<? extends T> buffer = null;
1474             }
1475         }
1476         class GenericDemo {
1477             public static <T> Iterator<T> collapse(Iterator<? extends T> source
1478                 ) {
1479                     throw new UnsupportedOperationException("Not supported");
1480                 }
1481             }
1482         }
1483         public void remove() {
1484             throw new UnsupportedOperationException();
1485         }
1486         public T next() {
1487             if (!hasNext()) {
1488                 throw new NoSuchElementException("No more elements");
1489             }
1490             return buffer.next();
1491         }
1492         public boolean hasNext() {
1493             while (source.hasNext()
1494                 && (null == buffer
1495                     || !buffer.hasNext())) {
1496
1497             }
1498             buffer = source.next();
1499             return buffer.hasNext();
1500         }
1501     }
1502     public static void main(String[] args) {
1503         List<String> li = new ArrayList<String>();
1504         li.add("foo");
1505         li.add("bar");
1506         li.add("bar");
1507         li.iterator();
1508         li = new ArrayList<String>();
1509         li.add("bar");
1510         li.add("qux");
1511         li.iterator();
1512         li.add("bar");
1513         li.add("qux");
1514         Iterator<String> i = collapse(li.iterator());
1515         while (i.hasNext()) {
1516             System.out.println(i.next());
1517         }
1518     }
1519     public static void foo() {
1520         System.out.println("aaarg");
1521     }
1522     public class beta {
1523         static Random rand = new Random();
1524         private static final double gammaInt = 0;
1525         double result = 0;
1526         for (int i = 1; i <= x; i++) {
1527             result += Math.log(rand.nextDouble());
1528         }
1529         return result;
1530     }
1531     public static double draw(int a, int b) {
1532         double result = 0;
1533         for (int i = 1; i <= x; i++) {
1534             result += Math.log(rand.nextDouble());
1535         }
1536         return result;
1537     }
1538     public static double draw(int a, int b) {
1539         public static final double gammaInt = 0;
1540         return gamma(a) / (gamma(a) + gamma(b));
1541     }
1542     public static void main(String[] args) {
1543         for (int i = 0; i < 100000; i++) {
1544             System.out.println(draw(2, 5));
1545         }
1546     }
1547     class Thing {
1548         public Thing() {
1549             Runtime.getRuntime().addShutdownHook(new Thread() {
1550                 public void run() {
1551                     public Thing() {
1552                         Runtime.getRuntime().addShutdownHook(new Thread() {
1553                             public void run() {
1554                             }
1555                         });
1556                     }
1557                 }
1558             });
1559         }
1560         public void flush() {
1561             /* do some deferred action */
1562         }
1563         public static void main(String[] args) {
1564             public static void main(String[] args) {
1565                 Thing t;
1566                 while (true) { t = new Thing(); }
1567             }
1568         }
1569         class crash {
1570             public static void main(String[] args) {
1571                 Object[] o = new Object[1];
1572                 Object[] o = new Object[1];
1573                 while (null != o) { o = new Object[1]; }
1574             }
1575         }
1576         public static void main(String[] args) {
1577             public static void main(String[] args) {
1578                 Object[] o = new Object[1];
1579                 Object[] o = new Object[1];
1580                 while (null != o) { o = new Object[1]; }
1581             }
1582         }
1583     }
1584 }
```

beta.java
23/39
~/tmp/
02/17/2011

beta.java
24/39
~/tmp/
02/17/2011

```
1569 }
1570 }
1571 class GenericDemo {
1572     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
1573     ) {
1574         return new Iterator<T>() {
1575             private Iterator<? extends T> buffer = null;
1576             public void remove() {
1577                 throw new UnsupportedOperationException("Not supported");
1578             }
1579         };
1580     }
1581     public T next() {
1582         if (!hasNext()) {
1583             throw new NoSuchElementException("No more elements");
1584         }
1585         return buffer.next();
1586     }
1587     public boolean hasNext() {
1588         while (source.hasNext()) {
1589             if (buffer == null) {
1590                 buffer = new Buffer();
1591             }
1592             if (buffer.hasNext()) {
1593                 return buffer.hasNext();
1594             }
1595         }
1596     }
1597     public void main(String[] args) {
1598         List<String> li = new ArrayList<String>();
1599         li.add("foo");
1600         li.add("bar");
1601         li.add(1, "baz");
1602         li.add(1, iterator());
1603         li = new ArrayList<String>();
1604         li.add("baz");
1605         li.add("qux");
1606         Iterator<String> i = collapse(li.iterator());
1607         while (!i.hasNext()) {
1608             System.out.println(i.next());
1609         }
1610     }
1611     public class Beta {
1612         static Random rand = new Random();
1613         public static void aaarg() {
1614             System.out.println("aaarg");
1615         }
1616     }
1617     class Foo {
1618         static Random rand = new Random();
1619         public static void aaarg() {
1620             System.out.println("aaarg");
1621         }
1622     }
1623     static Random rand = new Random();
1624     private static final double gamma() {
1625         double result = 0;
1626         for (int i = 1; i < x; i++) {
1627             result += Math.log(rand.nextDouble());
1628         }
1629     }
1630     return result;
1631 }
1632     static Random rand = new Random();
1633     public static double drawInt_a, int b) {
1634         return gamma() / (gamma(a) + gamma(b));
1635     }
1636 }
1637     public static void main(String[] args) {
1638         for (int i = 0; i < 100000; i++) {
1639             System.out.println(i);
1640             System.out.println(draw(2, 5));
1641         }
1642     }
1643 }
1644 class Thing {
1645     public Runtime() {
1646         Runtime.getRuntime().addShutdownHook(new Thread() {
1647             public void run() {
1648                 flush();
1649             }
1650         });
1651     }
1652     public void flush() {
1653         public static void main(String[] args) {
1654             public static void main(String[] args) {
1655                 Thing t;
1656                 while (true) {
1657                     t = new Thing();
1658                     Object[] o = new Object[1];
1659                     o[0] = new Object();
1660                     while (null != o[0]) {
1661                         o[0] = new Object();
1662                     }
1663                 }
1664             }
1665             class Crash {
1666                 public static void main(String[] args) {
1667                     Object[] o = new Object[1];
1668                     o[0] = new Object();
1669                     while (null != o[0]) {
1670                         o[0] = new Object();
1671                     }
1672                 }
1673             }
1674             class GenericDemo {
1675                 public static <T> Iterator<T> collapse(final Iterator<? extends T>> source
1676                 ) {
1677                     return new Iterator<T>() {
1678                         private Iterator<? extends T> buffer = null;
1679                         public void remove() {
1680                             throw new UnsupportedOperationException("Not supported");
1681                         }
1682                         public T next() {
1683                             if (!hasNext()) {
1684                                 throw new NoSuchElementException("No more elements");
1685                             }
1686                             public void main(String[] args) {
1687                                 List<String> li = new ArrayList<String>();
1688                                 li.add("foo");
1689                                 li.add("bar");
1690                                 li.add(1, iterator());
1691                                 li = new ArrayList<String>();
1692                                 li.add("baz");
1693                                 li.add("qux");
1694                                 Iterator<String> i = collapse(li.iterator());
1695                                 while (!i.hasNext()) {
1696                                     System.out.println(i.next());
1697                                 }
1698                                 }
1699                             }
1700                         }
1701                         public void main(String[] args) {
1702                             List<String> li = new ArrayList<String>();
1703                             li.add("foo");
1704                             li.add("bar");
1705                             li.add(1, iterator());
1706                             li = new ArrayList<String>();
1707                             li.add("baz");
1708                             li.add("qux");
1709                             Iterator<String> i = collapse(li.iterator());
1710                             while (!i.hasNext()) {
1711                                 System.out.println(i.next());
1712                             }
1713                         }
1714                     }
1715                 }
1716             }
1717             public static void main(String[] args) {
1718                 System.out.println("main");
1719             }
1720         }
1721     }
1722 }
```

beta.java
~ /tmp/

26/39
02/17/2011
beta.java
~/tmp/

```

782     buffer = source.next();
783 }
784
785     );
786 }
787 }
788
789 public static void main(String[] args) {
790     ListIterator<String> li = new ArrayList<String>();
791     List<String> l = new ArrayList<String>();
792     l.add("foo");
793     l.add("bar");
794     li.add(l.iterator());
795     li.add(1, iterator());
796     l = new ArrayList<String>();
797     l.add("baz");
798     l.add(iterator());
799     Iterator<String> i = collapse(l.iterator());
800     while (i.hasNext()) {
801         l.add(i.next());
802     }
803     System.out.println(l);
804 }
805
806 }
807
808 class foo {
809     public void aaarg() { System.out.println("aaargs"); }
810 }
811
812 public class beta {
813     static Random rand = new Random();
814
815     private static final double gam(int x) {
816         double result = 0;
817         for (int i = 1; i <= x; i++) {
818             result += Math.log(rand.nextDouble());
819         }
820     }
821     return result;
822 }
823
824 public static double draw(int a, int b) {
825     return gam(a) / (gam(a) + gam(b));
826 }
827
828 public static void main(String[] args) {
829     for (int i = 0; i < 100000; i++) {
830         System.out.println(draw(2, 5));
831     }
832 }
833
834 class Thing {
835     public Thing() {
836         Runtime.getRuntime().addShutdownHook(new Thread() {
837             public void run() { flush(); }
838         });
839     }
840 }
841
842 public void flush() /* do some deferred action */ {
843     Object o = new Object[1];
844     Object t = new Object[1];
845     while (true) { t = new Thing(); }
846 }
847
848 }
849
850 class crash {
851     public static void main(String[] args) {
852         Object o = new Object[1];
853         Object t = new Object[1];
854         while (null != o) { o = new Object[1]; }
855     }
856 }
```

beta.java
~/tmp/
27/39
02/17/2011

```
1854     }
1855 }
1856 class GenericDemo {
1857     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
1858     ) {
1859         return new Iterator<T>() {
1860             private Iterator<? extends T> buffer = null;
1861             public void remove() {
1862                 throw new UnsupportedOperationException("Not supported");
1863             }
1864             public T next() {
1865                 if (!hasNext()) {
1866                     throw new NoSuchElementException("No more elements");
1867                 }
1868                 return buffer.next();
1869             }
1870             public boolean hasNext() {
1871                 while (source.hasNext()) {
1872                     if (null == buffer)
1873                         buffer = source.next();
1874                     else
1875                         buffer = buffer.hasNext();
1876                 }
1877             }
1878             public T next() {
1879                 return buffer.hasNext();
1880             }
1881         };
1882     }
1883 }
1884 public static void main(String[] args) {
1885     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1886     List<String> l = new ArrayList<String>();
1887     l.add("foo");
1888     l.add("bar");
1889     li.add(l.iterator());
1890     l = new ArrayList<String>();
1891     l.add("baz");
1892     l.add("qux");
1893     li.add(l.iterator());
1894     Iterator<String> i = collapse(li.iterator());
1895     while (!i.hasNext()) {
1896         System.out.println(i.next());
1897     }
1898 }
1899 }
1900 }
1901 }
1902 class foo {
1903     public static void aaarg() {
1904         System.out.println("aaarg");
1905     }
1906 }
1907 public class beta {
1908     static Random rand = new Random();
1909     private static final double gamma(int x) {
1910         double result = 0;
1911         for (int i = 1; i <= x; i++) {
1912             result += Math.log(rand.nextDouble());
1913         }
1914     }
1915     public static void main(String[] args) {
1916         static Random rand = new Random();
1917         private static final double gamma(int x) {
1918             double result = 0;
1919             for (int i = 1; i <= x; i++) {
1920                 result += Math.log(rand.nextDouble());
1921             }
1922         }
1923         public static void main(String[] args) {
1924             for (int i = 0; i < 100000; i++) {
1925                 System.out.println(draw(2, 5));
1926             }
1927         }
1928     }
1929 }
1930 class Thing {
1931     public void run() {
1932         Runtime.getRuntime().addShutdownHook(new Thread() {
1933             public void run() {
1934                 flush();
1935             }
1936         });
1937     }
1938     public void flush() {
1939         System.out.println("flush()");
1940     }
1941     public static void main(String[] args) {
1942         Thing t = new Thing();
1943         t.run();
1944     }
1945     public static void main(String[] args) {
1946         Object[] o = new Object[1];
1947         o[0] = new Object();
1948         while (true) {
1949             t = new Thing();
1950             t.run();
1951             o[0] = new Object();
1952             while (true) {
1953                 t = new Thing();
1954                 t.run();
1955             }
1956             return new Iterator<T>() {
1957                 private Iterator<T> buffer = null;
1958                 public void remove() {
1959                     throw new UnsupportedOperationException("Not supported");
1960                 }
1961                 public T next() {
1962                     if (!hasNext())
1963                         throw new NoSuchElementException("No more elements");
1964                     return buffer.next();
1965                 }
1966                 public boolean hasNext() {
1967                     while (source.hasNext())
1968                         if (source.hasNext())
1969                             return true;
1970                     return false;
1971                 }
1972                 public void add(T element) {
1973                     if (null == buffer)
1974                         buffer = source.next();
1975                     else
1976                         buffer.add(element);
1977                 }
1978             };
1979         }
1980     }
1981     public static void main(String[] args) {
1982         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1983         List<String> l = new ArrayList<String>();
1984         l.add("foo");
1985         l.add("bar");
1986         li.add(l.iterator());
1987         public boolean hasNext() {
1988             while (source.hasNext())
1989                 if (source.hasNext())
1990                     return true;
1991                 return false;
1992             return false;
1993         }
1994         public void add(T element) {
1995             if (null == buffer)
1996                 buffer = source.next();
1997             else
1998                 buffer.add(element);
1999         }
2000     }
2001     public static void main(String[] args) {
2002         System.out.println("main()");
2003         System.out.println("main()");
2004         System.out.println("main()");
2005     }
2006 }
```

beta.java
~/tmp/
28/39
02/17/2011

```
1925     for (int i = 0; i < 100000; i++) {
1926         System.out.println(draw(2, 5));
1927     }
1928 }
1929 }
1930 class Thing {
1931     public void run() {
1932         Runtime.getRuntime().addShutdownHook(new Thread() {
1933             public void run() {
1934                 flush();
1935             }
1936         });
1937     }
1938     public void flush() {
1939         System.out.println("flush()");
1940     }
1941     public static void main(String[] args) {
1942         Thing t = new Thing();
1943         t.run();
1944     }
1945     public static void main(String[] args) {
1946         Object[] o = new Object[1];
1947         o[0] = new Object();
1948         while (true) {
1949             t = new Thing();
1950             t.run();
1951             o[0] = new Object();
1952             while (true) {
1953                 t = new Thing();
1954                 t.run();
1955             }
1956             return new Iterator<T>() {
1957                 private Iterator<T> buffer = null;
1958                 public void remove() {
1959                     throw new UnsupportedOperationException("Not supported");
1960                 }
1961                 public T next() {
1962                     if (!hasNext())
1963                         throw new NoSuchElementException("No more elements");
1964                     return buffer.next();
1965                 }
1966                 public boolean hasNext() {
1967                     while (source.hasNext())
1968                         if (source.hasNext())
1969                             return true;
1970                     return false;
1971                 }
1972                 public void add(T element) {
1973                     if (null == buffer)
1974                         buffer = source.next();
1975                     else
1976                         buffer.add(element);
1977                 }
1978             };
1979         }
1980     }
1981     public static void main(String[] args) {
1982         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1983         List<String> l = new ArrayList<String>();
1984         l.add("foo");
1985         l.add("bar");
1986         li.add(l.iterator());
1987         public boolean hasNext() {
1988             while (source.hasNext())
1989                 if (source.hasNext())
1990                     return true;
1991                 return false;
1992             return false;
1993         }
1994         public void add(T element) {
1995             if (null == buffer)
1996                 buffer = source.next();
1997             else
1998                 buffer.add(element);
1999         }
2000     }
2001     public static void main(String[] args) {
2002         System.out.println("main()");
2003         System.out.println("main()");
2004         System.out.println("main()");
2005     }
2006 }
```

```

beta.java          02/17/2011
~/tmp/           30/39

  } || !buffer.hasNext() ) {
    buffer = source.next();
  }
  return buffer.hasNext();
}

public static void main(String[] args) {
  List<Iterator<String>> li = new ArrayList<Iterator<String>>();
  li.add("foo");
  li.add("bar");
  li.add(iterator());
  li.add(new ArrayAdapter<String>());
  li.add("baz");
  li.add("qux");
  li.addAll(iterator());
  Iterator<String> i = collapse(li.iterator());
  while (i.hasNext()) {
    System.out.println(i.next());
  }
}

class foo {
  public static void aaarg() { System.out.println("aaarg"); }
}

public class beta {
  static Random rand = new Random();
  private static final double gam(int x) {
    double result = 0;
    for (int i = 1; i <= x; i++) {
      result += Math.log(rand.nextDouble());
    }
    return result;
  }
  public static double draw(int a, int b) {
    return gam(a) / (gam(a) + gam(b));
  }
  public static void main(String[] args) {
    for (int i = 0; i < 100000; i++) {
      System.out.println(draw(2, 5));
    }
  }
}

class Thing {
  public static void run() {
    Runtime.getRuntime().addShutdownHook(new Thread() {
      public void run() { flush(); }
    });
  }
  public void flush() { /* do some deferred action */ }
}

public static void main(String[] args) {
  Thing t;
  Object[] o = new Object[1];
  boolean (true) { t = new Thing(); }
}

class crash {
  public static void main(String[] args) {
    Object[] o = new Object[1];
  }
}

```

beta.java
~/tmp/
31/39
02/17/2011

```
2139     while (null != o) { o = new Object[]{o}; }
2140   }
2141 }
2142 class GenericDemo {
2143     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
2144     ) {
2145         return new Iterator<T>() {
2146             private Iterator<? extends T> buffer = null;
2147             public void remove() {
2148                 throw new UnsupportedOperationException("Not supported");
2149             }
2150             public T next() {
2151                 if (!hasNext()) {
2152                     throw new NoSuchElementException("No more elements");
2153                 }
2154                 return buffer.next();
2155             }
2156             public boolean hasNext() {
2157                 while (source.hasNext()) {
2158                     if (null == buffer)
2159                         buffer = source.next();
2160                     if (buffer.hasNext())
2161                         return true;
2162                     if (null == buffer)
2163                         buffer = source.next();
2164                 }
2165             }
2166             return buffer.hasNext();
2167         };
2168     }
2169 }
2170 public static void main(String[] args) {
2171     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2172     List<String> l1 = new ArrayList<String>();
2173     l1.add("foo");
2174     l1.add("bar");
2175     li.add(l1.iterator());
2176     l1 = new ArrayList<String>();
2177     l1.add("baz");
2178     l1.add("qux");
2179     li.add(l1.iterator());
2180     Iterator<String> i = collapse(li.iterator());
2181     while (!i.hasNext()) {
2182         System.out.println(i.next());
2183     }
2184 }
2185 }
2186 }
2187 }
2188 class foo {
2189     public static void aaarg() {
2190         System.out.println("aaarg");
2191     }
2192 }
2193 public class beta {
2194     static Random rand = new Random();
2195     private static final double gamin(int x) {
2196         double result = 0;
2197         for (int i = 1; i <= x; i++) {
2198             result += Math.log(rand.nextDouble());
2199         }
2200     }
2201     return result;
2202 }
2203 }
2204 static Random rand = new Random();
2205 private static final double gam(int x) {
2206     double result = 0;
2207     for (int i = 1; i <= x; i++) {
2208         result += Math.log(rand.nextDouble());
2209     }
2210 }
2211 public static void main(String[] args) {
2212     for (int i = 0; i < 100000; i++) {
2213         System.out.println(draw(2, 5));
2214     }
2215 }
2216 class Thing {
2217     public Thing() {
2218         Runtime.getRuntime().addShutdownHook(new Thread() {
2219             public void run() {
2220                 flush();
2221             }
2222         });
2223     }
2224     public void flush() {
2225         /* do some deferred action */
2226         Thing t = new Thing();
2227         while (true) {
2228             t.new Thing();
2229         }
2230     }
2231     public static void main(String[] args) {
2232         Object[] o = new Object[1];
2233         while (null != o) {
2234             o = new Object[1];
2235         }
2236     }
2237 }
2238 class GenericDemo {
2239     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
2240     ) {
2241         return new Iterator<T>() {
2242             private Iterator<? extends T> buffer = null;
2243             public void main(String[] args) {
2244                 Object[] o = new Object[1];
2245                 while (null != o) {
2246                     o = new Object[1];
2247                 }
2248             }
2249             public void hasNext() {
2250                 if (!hasNext())
2251                     throw new NoSuchElementException("No more elements");
2252             }
2253             public void next() {
2254                 if (!hasNext())
2255                     throw new NoSuchElementException("No more elements");
2256             }
2257             public void remove() {
2258                 throw new UnsupportedOperationException("Not supported");
2259             }
2260             public T next() {
2261                 if (!hasNext())
2262                     throw new NoSuchElementException("No more elements");
2263                 return buffer.next();
2264             }
2265             public void add(T element) {
2266                 if (null == buffer)
2267                     buffer = source.next();
2268                 else
2269                     buffer.add(element);
2270             }
2271             public void addAll(Collection<T> c) {
2272                 if (null == buffer)
2273                     buffer = source.next();
2274                 else
2275                     buffer.addAll(c);
2276             }
2277             public void clear() {
2278                 if (null == buffer)
2279                     buffer = source.next();
2280             }
2281         };
2282     }
2283     public static void main(String[] args) {
2284         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2285         List<String> l1 = new ArrayList<String>();
2286         l1.add("foo");
2287         l1.add("bar");
2288         li.add(l1.iterator());
2289         l1.add("baz");
2290         l1.add("qux");
2291         li.add(l1.iterator());
2292         Iterator<String> i = collapse(li.iterator());
2293         while (!i.hasNext()) {
2294             System.out.println(i.next());
2295         }
2296     }
2297 }
2298 
```

beta.java
~/tmp/
32/39
02/17/2011

```
2299     for (int i = 0; i < 100000; i++) {
2300         System.out.println(draw(2, 5));
2301     }
2302 }
2303 }
2304 class Thing {
2305     public Thing() {
2306         Runtime.getRuntime().addShutdownHook(new Thread() {
2307             public void run() {
2308                 flush();
2309             }
2310         });
2311     }
2312     public void flush() {
2313         /* do some deferred action */
2314         Thing t = new Thing();
2315         while (true) {
2316             t.new Thing();
2317         }
2318     }
2319 }
2320 
```

33/39
02/17/2011
beta.java
~/tmp/

beta.java
~/tmp/

34/39
02/17/2011
beta.java
~/tmp/p/

beta.java
35/39
~/tmp/
02/17/2011

```
2424     while (null != o) { o = new Object[]{o}; }
2425   }
2426 }
2427
2428 class GenericDemo {
2429   public static <T> Iterator<T> collapse(final Iterator<? extends T> source
2430   ) {
2431     return new Iterator<T>() {
2432       private Iterator<? extends T> buffer = null;
2433       public void remove() {
2434         throw new UnsupportedOperationException("Not supported");
2435       }
2436       public T next() {
2437         if (!hasNext()) {
2438           throw new NoSuchElementException("No more elements");
2439         }
2440         return buffer.next();
2441       }
2442       public boolean hasNext() {
2443         while (source.hasNext()) {
2444           if (null == buffer)
2445             buffer = new ArrayList();
2446           buffer.add(source.next());
2447         }
2448       }
2449       public void collapse() {
2450         Iterator<String> i = new ArrayList<Iterator<String>>();
2451         i.add("foo");
2452         i.add(1.iterator());
2453         i = new ArrayList<String>();
2454         i.add("baz");
2455         i.add("qux");
2456         i.add(1.iterator());
2457         Iterator<String> i = collapse(i.iterator());
2458         while (!i.hasNext()) {
2459           System.out.println(i.next());
2460         }
2461       }
2462       public class beta {
2463         static Random rand = new Random();
2464         private static final double gamma(int x) {
2465           double result = 0;
2466           for (int i = 1; i <= x; i++) {
2467             result += Math.log(rand.nextDouble());
2468           }
2469         }
2470       }
2471     }
2472   }
2473   class foo {
2474     public static void aaarg() { System.out.println("aaarg"); }
2475   }
2476 }
2477
2478 Public class beta {
2479   static Random rand = new Random();
2480   private static final double gamma(int x) {
2481     double result = 0;
2482     for (int i = 1; i <= x; i++) {
2483       result += Math.log(rand.nextDouble());
2484     }
2485   }
2486 }
```

beta.java
36/39
~/tmp/
02/17/2011

```
2487   for (int i = 0; i < 100000; i++) {
2488     System.out.println(draw(2, 5));
2489   }
2490 }
2491
2492 class Thing {
2493   public Thing() {
2494     Runtime.getRuntime().addShutdownHook(new Thread() {
2495       public void run() { flush(); }
2496     });
2497   }
2498 }
2499
2500 public void flush() { /* do some deferred action */ }
2501
2502 public static void main(String[] args) {
2503   Thing t = new Thing();
2504   t.flush();
2505 }
2506
2507 public static void main(String[] args) {
2508   Thing t = new Thing();
2509   t.flush();
2510 }
2511
2512 public static void main(String[] args) {
2513   Object[] o = new Object[1];
2514   o[0] = new Object();
2515   while (true) { o[0] = new Object[1][o]; }
2516 }
2517
2518 class crash {
2519   public static void main(String[] args) {
2520     Object[] o = new Object[1];
2521     o[0] = new Object[1];
2522     while (true) { o[0] = new Object[1][o]; }
2523 }
2524
2525 class GenericDemo {
2526   public static <T> Iterator<T> collapse(final Iterator<? extends T> source
2527   ) {
2528     return new Iterator<T>() {
2529       private Iterator<? extends T> buffer = null;
2530       public void remove() {
2531         throw new UnsupportedOperationException("Not supported");
2532       }
2533       public T next() {
2534         if (!hasNext()) {
2535           throw new NoSuchElementException("No more elements");
2536         }
2537         return buffer.next();
2538       }
2539       public boolean hasNext() {
2540         if (!hasNext())
2541           throw new NoSuchElementException("No more elements");
2542         return buffer.hasNext();
2543       }
2544       public void collapse() {
2545         Iterator<String> i = collapse(i.iterator());
2546         while (!i.hasNext()) {
2547           System.out.println(i.next());
2548         }
2549       }
2550     };
2551   }
2552 }
```

beta.java
~/tmp/

38/39
02/17/2011
beta.java
~/{tmp/}

38/39
02/17/2011
beta.java
~/{tmp}/

beta.java39/39
~/tmp/

02/17/2011

```
2700     while (null != o) { o = new Object()[0]; }
2710   }
2711 }
2712 class GenericDemo {
2713   public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
2714   ) {
2715     return new Iterator<T>() {
2716       private Iterator<? extends T> buffer = null;
2717       public void remove() {
2718         throw new UnsupportedOperationException("Not supported");
2719       }
2720     };
2721   }
2722   public T next() {
2723     if (!hasNext()) {
2724       throw new NoSuchElementException("No more elements");
2725     }
2726     return buffer.next();
2727   }
2728   public boolean hasNext() {
2729     while (source.hasNext()) {
2730       if (null == buffer)
2731         buffer = source.next();
2732       else if (!buffer.hasNext())
2733         buffer = source.next();
2734     }
2735     return buffer.hasNext();
2736   }
2737 }
2738 }
2739 }
2740 public static void main(String[] args) {
2741   List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2742   List<String> l = new ArrayList<String>();
2743   l.add("foo");
2744   l.add("bar");
2745   li.add(l.iterator());
2746   l = new ArrayList<String>();
2747   l.add("baz");
2748   l.add("quux");
2749   li.add(l.iterator());
2750   Iterator<String> i = collapse(li.iterator());
2751   while (i.hasNext()) {
2752     System.out.println(i.next());
2753   }
2754 }
2755 }
2756 }
2757 }
2758 }
```