

```

1 import java.util.Random;
2 import java.util.*;
3 class foo {
4     public static void aaarg() { System.out.println("aaarg"); }
5 }
6
7 public class beta {
8     static Random rand = new Random();
9
10    private static final double gam(int x) {
11        double result = 0;
12        for (int i = 1; i <= x; i++) {
13            result += Math.log(rand.nextDouble());
14        }
15        return result;
16    }
17
18    public static double draw(int a, int b) {
19        return gam(a) / (gam(a) + gam(b));
20    }
21
22    public static void main(String[] args) {
23        for (int i = 0; i < 100000; i++) {
24            System.out.println(draw(2, 5));
25        }
26    }
27 }
28
29 class Thing {
30     public Thing() {
31         Runtime.getRuntime().addShutdownHook(new Thread() {
32             public void run() { flush(); }
33         });
34     }
35
36     public void flush() { /* do some deferred action */ }
37
38     public static void main(String[] args) {
39         Thing t;
40         while (true) { t = new Thing(); }
41     }
42 }
43
44 class crash {
45     public static void main(String[] args) {
46         Object[] o = new Object[10];
47         while (null != o) { o = new Object[10]; }
48     }
49 }
50
51 class GenericDemo {
52     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
53         return new Iterator<T>() {
54             private Iterator<? extends T> buffer = null;
55             public void remove() {
56                 throw new UnsupportedOperationException("Not supported");
57             }
58             public T next() {
59                 if (buffer == null) {
60                     throw new NoSuchElementException("No more elements");
61                 }
62                 return buffer.next();
63             }
64             public boolean hasNext() {
65                 while (source.hasNext()) {
66                     // ...
67                 }
68             }
69         }
70     }
71 }

```

```

72     }
73     }
74     }
75     }
76     }
77     }
78     }
79     }
80     }
81     }
82     }
83     }
84     }
85     }
86     }
87     }
88     }
89     }
90     }
91     }
92     }
93     }
94     }
95     }
96     }
97     }
98     }
99     }
100    }
101    }
102    }
103    }
104    }
105    }
106    }
107    }
108    }
109    }
110    }
111    }
112    }
113    }
114    }
115    }
116    }
117    }
118    }
119    }
120    }
121    }
122    }
123    }
124    }
125    }
126    }
127    }
128    }
129    }
130    }
131    }
132    }
133    }
134    }
135    }
136    }
137    }
138    }
139    }
140    }
141    }
142    }
143    }
144    }

```

```

144     } while (null != o) { o = new Object[o]; }
145     }
146     }
147     }
148     class GenericDemo {
149     } {
150     public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<? extends T>> source
151     private Iterator<T> extends T; buffer = null;
152     public void remove() {
153     } throw new UnsupportedOperationException("Not supported");
154     }
155     } throw new UnsupportedOperationException("Not supported");
156     }
157     public T next() {
158     if (!hasNext()) {
159     } throw new NoSuchElementException("No more elements");
160     } return buffer.next();
161     }
162     }
163     }
164     }
165     }
166     }
167     }
168     }
169     }
170     }
171     }
172     }
173     }
174     }
175     }
176     }
177     }
178     }
179     }
180     }
181     }
182     }
183     }
184     }
185     }
186     }
187     }
188     }
189     }
190     }
191     }
192     }
193     }
194     }
195     }
196     }
197     }
198     }
199     }
200     }
201     }
202     }
203     }
204     }
205     }
206     }
207     }
208     }
209     }
210     }
211     }
212     }
213     }
214     }

```

```

215     for (int i = 0; i < 100000; i++) {
216     System.out.println(draw(2, 5));
217     }
218     }
219     }
220     }
221     }
222     }
223     }
224     }
225     }
226     }
227     }
228     }
229     }
230     }
231     }
232     }
233     }
234     }
235     }
236     }
237     }
238     }
239     }
240     }
241     }
242     }
243     }
244     }
245     }
246     }
247     }
248     }
249     }
250     }
251     }
252     }
253     }
254     }
255     }
256     }
257     }
258     }
259     }
260     }
261     }
262     }
263     }
264     }
265     }
266     }
267     }
268     }
269     }
270     }
271     }
272     }
273     }
274     }
275     }
276     }
277     }
278     }
279     }
280     }
281     }
282     }
283     }
284     }
285     }

```

```
286 }
287
288
289
290 class foo {
291     public static void aaarg() { System.out.println("aaarg"); }
292 }
293
294 public class beta {
295     static Random rand = new Random();
296
297     private static final double gam(int x) {
298         double result = 0;
299         for (int i = 1; i <= x; i++) {
300             result += Math.log(rand.nextDouble());
301         }
302         return result;
303     }
304
305     public static double draw(int a, int b) {
306         return gam(a) / (gam(a) + gam(b));
307     }
308
309     public static void main(String[] args) {
310         for (int i = 0; i < 100000; i++) {
311             System.out.println(draw(2, 5));
312         }
313     }
314 }
315
316
317
318 class Thing {
319     public Thing() {
320         Runtime.getRuntime().addShutdownHook(new Thread() {
321             public void run() { flush(); }
322         });
323     }
324
325     public void flush() { /* do some deferred action */ }
326
327     public static void main(String[] args) {
328         Thing t;
329         while (true) { t = new Thing(); }
330     }
331 }
332
333
334
335 class crash {
336     public static void main(String[] args) {
337         Object[] o = new Object[10];
338         while (null != o) { o = new Object[10]; }
339     }
340 }
341
342
343
344 class GenericDemo {
345     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
346         return new Iterator<T>() {
347             private Iterator<? extends T> buffer = null;
348
349             public void remove() {
350                 throw new UnsupportedOperationException("Not supported");
351             }
352
353             public T next() {
354                 if (!hasNext()) {
355                     throw new NoSuchElementException("No more elements");
356                 }
357                 return buffer.next();
358             }
359
360             public boolean hasNext() {
361                 while (source.hasNext())
362                     && (null == buffer
363
```

```
357         || !buffer.hasNext()) {
358             buffer = source.next();
359         }
360         return buffer.hasNext();
361     }
362 }
363
364
365
366 public static void main(String[] args) {
367     List<String> l = new ArrayList<String>();
368     List<String> l1 = new ArrayList<String>();
369     l.add("foo"); l.add("bar");
370     l1.add(l.iterator());
371
372     l = new ArrayList<String>();
373     l.add("baz"); l.add("quax");
374     l1.add(l.iterator());
375
376     Iterator<String> i = collapse(l1.iterator());
377     while (i.hasNext()) {
378         System.out.println(i.next());
379     }
380 }
381
382
383
384 class foo {
385     public static void aaarg() { System.out.println("aaarg"); }
386 }
387
388
389
390 public class beta {
391     static Random rand = new Random();
392
393     private static final double gam(int x) {
394         double result = 0;
395         for (int i = 1; i <= x; i++) {
396             result += Math.log(rand.nextDouble());
397         }
398         return result;
399     }
400
401     public static double draw(int a, int b) {
402         return gam(a) / (gam(a) + gam(b));
403     }
404
405     public static void main(String[] args) {
406         for (int i = 0; i < 100000; i++) {
407             System.out.println(draw(2, 5));
408         }
409     }
410 }
411
412
413
414 class Thing {
415     Runtime.getRuntime().addShutdownHook(new Thread() {
416         public void run() { flush(); }
417     });
418
419     public void flush() { /* do some deferred action */ }
420
421     public static void main(String[] args) {
422         Thing t;
423         while (true) { t = new Thing(); }
424     }
425 }
426
427
428
429 class crash {
430     public static void main(String[] args) {
431         Object[] o = new Object[10];
432
```

```

429     } while (null != o) { o = new Object[o]; }
430     }
431     }
432     }
433     class GenericDemo {
434     public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<? extends T>> source
435     ) {
436     return new Iterator<T>() {
437     private Iterator<? extends T> buffer = null;
438     public void remove() {
439     throw new UnsupportedOperationException("Not supported");
440     }
441     public T next() {
442     if (!hasNext()) {
443     throw new NoSuchElementException("No more elements");
444     }
445     return buffer.next();
446     }
447     public boolean hasNext() {
448     while (source.hasNext()) {
449     Object o = source.next();
450     if (null != o) {
451     buffer = source.next();
452     }
453     }
454     return buffer.hasNext();
455     }
456     }
457     }
458     }
459     }
460     }
461     public static void main(String[] args) {
462     List<String> li = new ArrayList<String>();
463     li.add("foo"); li.add("bar");
464     li.add("foo");
465     li.add("bar");
466     li.add("foo");
467     li.add("bar");
468     li.add("foo");
469     li.add("bar");
470     li.add("foo");
471     li.add("bar");
472     li.add("foo");
473     li.add("bar");
474     li.add("foo");
475     li.add("bar");
476     li.add("foo");
477     li.add("bar");
478     li.add("foo");
479     li.add("bar");
480     li.add("foo");
481     li.add("bar");
482     li.add("foo");
483     li.add("bar");
484     li.add("foo");
485     li.add("bar");
486     li.add("foo");
487     li.add("bar");
488     li.add("foo");
489     li.add("bar");
490     li.add("foo");
491     li.add("bar");
492     li.add("foo");
493     li.add("bar");
494     li.add("foo");
495     li.add("bar");
496     li.add("foo");
497     li.add("bar");
498     li.add("foo");
499     li.add("bar");

```

```

500     for (int i = 0; i < 100000; i++) {
501     System.out.println(draw(2, 5));
502     }
503     }
504     }
505     }
506     class Thing {
507     public Thing() {
508     Runtime.getRuntime().addShutdownHook(new Thread() {
509     public void run() { flush(); }
510     });
511     }
512     public void flush() { /* do some deferred action */ }
513     }
514     public static void main(String[] args) {
515     Thing t;
516     while (true) { t = new Thing(); }
517     }
518     }
519     }
520     }
521     class crash {
522     public static void main(String[] args) {
523     Object[] o = new Object[1];
524     while (null != o) { o = new Object[o]; }
525     }
526     }
527     }
528     class GenericDemo {
529     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
530     ) {
531     return new Iterator<T>() {
532     private Iterator<? extends T> buffer = null;
533     public void remove() {
534     throw new UnsupportedOperationException("Not supported");
535     }
536     public T next() {
537     if (!hasNext()) {
538     throw new NoSuchElementException("No more elements");
539     }
540     return buffer.next();
541     }
542     }
543     }
544     }
545     }
546     }
547     }
548     }
549     }
550     }
551     }
552     }
553     }
554     }
555     }
556     }
557     }
558     }
559     }
560     }
561     }
562     }
563     }
564     }
565     }
566     }
567     }
568     }
569     }

```

```
571 }
572
573 class foo {
574     public static void aaarg() { System.out.println("aaarg"); }
575 }
576
577 public class beta {
578
579     static Random rand = new Random();
580
581     private static final double gam(int x) {
582         double result = 0;
583         for (int i = 1; i <= x; i++) {
584             result += Math.log(rand.nextDouble());
585         }
586         return result;
587     }
588
589     public static double draw(int a, int b) {
590         return gam(a) / (gam(a) + gam(b));
591     }
592
593     public static void main(String[] args) {
594         for (int i = 1; i <= 5; i++) {
595             System.out.println(draw(2, 5));
596         }
597     }
598 }
599
600 class Thing {
601     public Thing() {
602         Runtime.getRuntime().addShutdownHook(new Thread() {
603             public void run() { flush(); }
604         });
605     }
606
607     public void flush() { /* do some deferred action */ }
608
609     public static void main(String[] args) {
610         Thing t;
611         while (true) { t = new Thing(); }
612     }
613 }
614
615 class crash {
616     public static void main(String[] args) {
617         Object[] o = new Object[10];
618         while (null != o) { o = new Object[10]; }
619     }
620 }
621
622 class GenericDemo {
623     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
624         return new Iterator<T>() {
625             private Iterator<? extends T> buffer = null;
626
627             public void remove() {
628                 throw new UnsupportedOperationException("Not supported");
629             }
630
631             public T next() {
632                 if (!hasNext()) {
633                     throw new NoSuchElementException("No more elements");
634                 }
635                 return buffer.next();
636             }
637
638             public boolean hasNext() {
639                 while (source.hasNext())
640                     && (null == buffer)
641 }
```

```
642     || !buffer.hasNext()) {
643         buffer = source.next();
644     }
645     return buffer.hasNext();
646 }
647
648 public static void main(String[] args) {
649     List<String> l = new ArrayList<String>();
650     List<String> l2 = new ArrayList<String>();
651     l.add("foo"); l.add("bar");
652     l2.add(l.iterator());
653
654     l = new ArrayList<String>();
655     l.add("baz"); l.add("quax");
656     l2.add(l.iterator());
657
658     Iterator<String> i = collapse(l1.iterator());
659     while (i.hasNext()) {
660         System.out.println(i.next());
661     }
662 }
663
664 class foo {
665     public static void aaarg() { System.out.println("aaarg"); }
666 }
667
668 public class beta {
669     static Random rand = new Random();
670
671     private static final double gam(int x) {
672         double result = 0;
673         for (int i = 1; i <= x; i++) {
674             result += Math.log(rand.nextDouble());
675         }
676         return result;
677     }
678
679     public static double draw(int a, int b) {
680         return gam(a) / (gam(a) + gam(b));
681     }
682
683     public static void main(String[] args) {
684         for (int i = 0; i < 100000; i++) {
685             System.out.println(draw(2, 5));
686         }
687     }
688 }
689
690 class Thing {
691     public Thing() {
692         Runtime.getRuntime().addShutdownHook(new Thread() {
693             public void run() { flush(); }
694         });
695     }
696
697     public void flush() { /* do some deferred action */ }
698
699     public static void main(String[] args) {
700         Thing t;
701         while (true) { t = new Thing(); }
702     }
703 }
704
705 class crash {
706     public static void main(String[] args) {
707         Object[] o = new Object[10];
708         while (null != o) { o = new Object[10]; }
709     }
710 }
```

```

714 }
715 }
716 }
717 class GenericDemo {
718 } {
719     return new Iterator<T>() {
720     private Iterator<T> extends T; buffer = null;
721     public void remove() {
722     throw new UnsupportedOperationException("Not supported");
723     }
724     public T next() {
725     if (!hasNext()) {
726     throw new NoSuchElementException("No more elements");
727     }
728     return buffer.next();
729     }
730     public boolean hasNext() {
731     while (source.hasNext()
732     && (null == buffer
733     || !buffer.hasNext())) {
734     buffer = source.next();
735     }
736     return buffer.hasNext();
737     }
738     }
739     }
740     }
741     }
742     }
743     }
744     }
745     }
746     }
747     }
748     }
749     }
750     }
751     }
752     }
753     }
754     }
755     }
756     }
757     }
758     }
759     }
760     }
761     }
762     }
763     }
764     }
765     }
766     }
767     }
768     }
769     }
770     }
771     }
772     }
773     }
774     }
775     }
776     }
777     }
778     }
779     }
780     }
781     }
782     }
783     }
784     }

```

```

785     }
786     }
787     }
788     }
789     }
790     }
791     }
792     }
793     }
794     }
795     }
796     }
797     }
798     }
799     }
800     }
801     }
802     }
803     }
804     }
805     }
806     }
807     }
808     }
809     }
810     }
811     }
812     }
813     }
814     }
815     }
816     }
817     }
818     }
819     }
820     }
821     }
822     }
823     }
824     }
825     }
826     }
827     }
828     }
829     }
830     }
831     }
832     }
833     }
834     }
835     }
836     }
837     }
838     }
839     }
840     }
841     }
842     }
843     }
844     }
845     }
846     }
847     }
848     }
849     }
850     }
851     }
852     }
853     }
854     }
855     }

```

```
856 class foo {  
857     public static void aaarg() { System.out.println("aaarg"); }  
858 }  
859  
860 public class beta {  
861     static Random rand = new Random();  
862     private static final double gam(int x) {  
863         double result = 0;  
864         for (int i = 1; i <= x; i++) {  
865             result += Math.log(rand.nextDouble());  
866         }  
867         return result;  
868     }  
869     public static double draw(int a, int b) {  
870         return gam(a) / (gam(a) + gam(b));  
871     }  
872     public static void main(String[] args) {  
873         for (int i = 0; i < 100000; i++) {  
874             System.out.println(draw(2, 3));  
875         }  
876     }  
877  
878     class Thing {  
879         public Thing() {  
880             Runtime.getRuntime().addShutdownHook(new Thread() {  
881                 @Override public void run() { flush(); }  
882             });  
883         }  
884     }  
885  
886     public void flush() { /* do some deferred action */ }  
887     public static void main(String[] args) {  
888         Thing t;  
889         while (true) { t = new Thing(); }  
890     }  
891  
892     class crash {  
893         public void main(String[] args) {  
894             Object[] o = new Object[1];  
895             while (null != o) { o = new Object[1](o); }  
896         }  
897     }  
898  
899     class GenericDemo {  
900         public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source  
901         ) {  
902             return new Iterator<T>() {  
903                 private Iterator<? extends T> buffer = null;  
904                 public void remove() {  
905                     throw new UnsupportedOperationException("Not supported");  
906                 }  
907                 public T next() {  
908                     if (hasNext()) {  
909                         throw new NoSuchElementException("No more elements");  
910                     }  
911                     return buffer.next();  
912                 }  
913                 public boolean hasNext() {  
914                     while (source.hasNext())  
915                         && (null == buffer  
916                             || !buffer.hasNext()); }  
917             }  
918         }  
919     }  
920  
921     class crash {  
922         public static void main(String[] args) {  
923             Object[] o = new Object[1];  
924             while (null != o) { o = new Object[1](o); }  
925         }  
926     }  
927 }  
928 }  
929  
930     public static void main(String[] args) {  
931         List<String> l = new ArrayList<String>();  
932         l.add("foo"); l.add("bar");  
933         l.add(l.iterator());  
934  
935         l = new ArrayList<String>();  
936         l.add("baz"); l.add("quax");  
937         l.add(l.iterator());  
938  
939         Iterator<String> i = collapse(l.iterator());  
940         while (i.hasNext()) {  
941             System.out.println(i.next());  
942         }  
943     }  
944  
945     class foo {  
946         public static void aaarg() { System.out.println("aaarg"); }  
947     }  
948  
949     public class beta {  
950         static Random rand = new Random();  
951         private static final double gam(int x) {  
952             double result = 0;  
953             for (int i = 1; i <= x; i++) {  
954                 result += Math.log(rand.nextDouble());  
955             }  
956             return result;  
957         }  
958         public static double draw(int a, int b) {  
959             return gam(a) / (gam(a) + gam(b));  
960         }  
961         public static void main(String[] args) {  
962             for (int i = 0; i < 100000; i++) {  
963                 System.out.println(draw(2, 5));  
964             }  
965         }  
966     }  
967  
968     class Thing {  
969         public Thing() {  
970             Runtime.getRuntime().addShutdownHook(new Thread() {  
971                 @Override public void run() { flush(); }  
972             });  
973         }  
974         public void flush() { /* do some deferred action */ }  
975         public static void main(String[] args) {  
976             Thing t;  
977             while (true) { t = new Thing(); }  
978         }  
979     }  
980  
981     class crash {  
982         public static void main(String[] args) {  
983             Object[] o = new Object[1];  
984             while (null != o) { o = new Object[1](o); }  
985         }  
986     }  
987 }  
988 }
```

```
927     }  
928     buffer = source.next();  
929 }  
930     return buffer.hasNext();  
931 }  
932 }  
933  
934     public static void main(String[] args) {  
935         List<String> l = new ArrayList<String>();  
936         l.add("foo"); l.add("bar");  
937         l.add(l.iterator());  
938  
939         l = new ArrayList<String>();  
940         l.add("baz"); l.add("quax");  
941         l.add(l.iterator());  
942  
943         Iterator<String> i = collapse(l.iterator());  
944         while (i.hasNext()) {  
945             System.out.println(i.next());  
946         }  
947     }  
948  
949     class foo {  
950         public static void aaarg() { System.out.println("aaarg"); }  
951     }  
952  
953     public class beta {  
954         static Random rand = new Random();  
955         private static final double gam(int x) {  
956             double result = 0;  
957             for (int i = 1; i <= x; i++) {  
958                 result += Math.log(rand.nextDouble());  
959             }  
960             return result;  
961         }  
962         public static double draw(int a, int b) {  
963             return gam(a) / (gam(a) + gam(b));  
964         }  
965         public static void main(String[] args) {  
966             for (int i = 0; i < 100000; i++) {  
967                 System.out.println(draw(2, 5));  
968             }  
969         }  
970     }  
971  
972     class Thing {  
973         public Thing() {  
974             Runtime.getRuntime().addShutdownHook(new Thread() {  
975                 @Override public void run() { flush(); }  
976             });  
977         }  
978         public void flush() { /* do some deferred action */ }  
979         public static void main(String[] args) {  
980             Thing t;  
981             while (true) { t = new Thing(); }  
982         }  
983     }  
984  
985     class crash {  
986         public static void main(String[] args) {  
987             Object[] o = new Object[1];  
988             while (null != o) { o = new Object[1](o); }  
989         }  
990     }  
991 }  
992 }
```

```

999 }
1000 }
1001 }
1002 }
1003 }
1004 }
1005 }
1006 }
1007 }
1008 }
1009 }
1010 }
1011 }
1012 }
1013 }
1014 }
1015 }
1016 }
1017 }
1018 }
1019 }
1020 }
1021 }
1022 }
1023 }
1024 }
1025 }
1026 }
1027 }
1028 }
1029 }
1030 }
1031 }
1032 }
1033 }
1034 }
1035 }
1036 }
1037 }
1038 }
1039 }
1040 }
1041 }
1042 }
1043 }
1044 }
1045 }
1046 }
1047 }
1048 }
1049 }
1050 }
1051 }
1052 }
1053 }
1054 }
1055 }
1056 }
1057 }
1058 }
1059 }
1060 }
1061 }
1062 }
1063 }
1064 }
1065 }
1066 }
1067 }
1068 }
1069 }

```

```

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 (!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<Iterator<String>> li = new ArrayList<Iterator<String>>();
        li.add("foo"); li.add("bar");
        li.add(li.iterator());
        li = new ArrayList<String>();
        li.add("baz"); li.add("quux");
        li.add(li.iterator());
        Iterator<String> i = collapse(li.iterator());
        while (i.hasNext()) {
            System.out.println(i.next());
        }
    }
}

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

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++) {

```

```

1070 }
1071 }
1072 }
1073 }
1074 }
1075 }
1076 }
1077 }
1078 }
1079 }
1080 }
1081 }
1082 }
1083 }
1084 }
1085 }
1086 }
1087 }
1088 }
1089 }
1090 }
1091 }
1092 }
1093 }
1094 }
1095 }
1096 }
1097 }
1098 }
1099 }
1100 }
1101 }
1102 }
1103 }
1104 }
1105 }
1106 }
1107 }
1108 }
1109 }
1110 }
1111 }
1112 }
1113 }
1114 }
1115 }
1116 }
1117 }
1118 }
1119 }
1120 }
1121 }
1122 }
1123 }
1124 }
1125 }
1126 }
1127 }
1128 }
1129 }
1130 }
1131 }
1132 }
1133 }
1134 }
1135 }
1136 }
1137 }
1138 }
1139 }
1140 }

```

```

        System.out.println(draw(2, 5));
    }
}

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[]();
        while (null != o) { o = new Object[]{}; }
    }
}

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 (!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<Iterator<String>> li = new ArrayList<Iterator<String>>();
        li.add("foo"); li.add("bar");
        li.add(li.iterator());
        li = new ArrayList<String>();
        li.add("baz"); li.add("quux");
        li.add(li.iterator());
        Iterator<String> i = collapse(li.iterator());
        while (i.hasNext()) {
            System.out.println(i.next());
        }
    }
}

```



```
1141 class foo {
1142     public static void aaarg() { System.out.println("aaarg"); }
1143 }
1144
1145 public class beta {
1146     static Random rand = new Random();
1147     private static final double gam(int x) {
1148         double result = 0;
1149         for (int i = 1; i <= x; i++) {
1150             result += Math.log(rand.nextDouble());
1151         }
1152         return result;
1153     }
1154     public static double draw(int a, int b) {
1155         return gam(a) / (gam(a) + gam(b));
1156     }
1157     public static void main(String[] args) {
1158         for (int i = 0; i < 100000; i++) {
1159             System.out.println(draw(2, 3));
1160         }
1161     }
1162 }
1163
1164 class Thing {
1165     public Thing() {
1166         Runtime.getRuntime().addShutdownHook(new Thread() {
1167             public void run() { flush(); }
1168         });
1169     }
1170     public void flush() { /* do some deferred action */ }
1171     public static void main(String[] args) {
1172         Thing t;
1173         while (true) { t = new Thing(); }
1174     }
1175 }
1176
1177 class crash {
1178     public static void main(String[] args) {
1179         Object[] o = new Object[1];
1180         while (null != o) { o = new Object[1](o); }
1181     }
1182 }
1183
1184 class GenericDemo {
1185     public static <T> Iterator<T> collapse(final Iterator<? extends T> source
1186     ) {
1187         return new Iterator<T>() {
1188             private Iterator<? extends T> buffer = null;
1189             public void remove() {
1190                 throw new UnsupportedOperationException("Not supported");
1191             }
1192             public T next() {
1193                 if (buffer == null) {
1194                     throw new NoSuchElementException("No more elements");
1195                 }
1196                 return buffer.next();
1197             }
1198             public boolean hasNext() {
1199                 while (source.hasNext()
1200                     && (null == buffer
1201                         || !buffer.hasNext())) {
1202                 }
1203             }
1204         }
1205     }
1206 }
```

```
212     }
213     buffer = source.next();
214 }
215     return buffer.hasNext();
216 }
217 }
218 }
219
220 public static void main(String[] args) {
221     List<String> strings = new ArrayList<String>();
222     List<String> list = new ArrayList<String>();
223     li.add("foo"); li.add("bar");
224     li.add(list.iterator());
225     l = new ArrayList<String>();
226     l.add("baz"); l.add("quax");
227     li.add(list.iterator());
228 }
229
230 Iterator<String> i = collapse(list.iterator());
231 while (i.hasNext()) {
232     System.out.println(i.next());
233 }
234 }
235 }
236
237 class foo {
238     public static void aaarg() { System.out.println("aaarg"); }
239 }
240
241 public class beta {
242     static Random rand = new Random();
243     private static final double gam(int x) {
244         double result = 0;
245         for (int i = 1; i <= x; i++) {
246             result += Math.log(rand.nextDouble());
247         }
248         return result;
249     }
250     public static double draw(int a, int b) {
251         return gam(a) / (gam(a) + gam(b));
252     }
253     public static void main(String[] args) {
254         for (int i = 0; i < 100000; i++) {
255             System.out.println(draw(2, 5));
256         }
257     }
258 }
259
260 class Thing {
261     public Thing() {
262         Runtime.getRuntime().addShutdownHook(new Thread() {
263             public void run() { flush(); }
264         });
265     }
266     public void flush() { /* do some deferred action */ }
267     public static void main(String[] args) {
268         Thing t;
269         while (true) { t = new Thing(); }
270     }
271 }
272
273 class crash {
274     public static void main(String[] args) {
275         Object[] o = new Object[1];
276         while (null != o) { o = new Object[1](o); }
277     }
278 }
```

```
1284 }
1285 }
1286 }
1287 }
1288 }
1289 }
1290 }
1291 }
1292 }
1293 }
1294 }
1295 }
1296 }
1297 }
1298 }
1299 }
1300 }
1301 }
1302 }
1303 }
1304 }
1305 }
1306 }
1307 }
1308 }
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 }

class GenericDemo {
    private Iterator<T> extends T; buffer = null;

    public void remove() {
        throw new UnsupportedOperationException("Not supported");
    }

    public T next() {
        if (!hasNext()) {
            throw new NoSuchElementException("No more elements");
        }
        return buffer.next();
    }

    public boolean hasNext() {
        while (source.hasNext())
            && (null != buffer
                && 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(li.iterator());

    l = new ArrayList<String>();
    l.add("baz"); l.add("quux");
    li.add(li.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++) {

```

```
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 }
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 }

System.out.println(draw(2, 5));

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[1][o]; }
    }
}

class GenericDemo {
    public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<T> source
    ) {
        return new Iterator<T>() {
            private Iterator<T> extends T; buffer = null;

            public void remove() {
                throw new UnsupportedOperationException("Not supported");
            }

            public T next() {
                if (!hasNext()) {
                    throw new NoSuchElementException("No more elements");
                }
                return buffer.next();
            }

            public boolean hasNext() {
                while (source.hasNext())
                    && (null != buffer
                        && 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(li.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(li.iterator());

        Iterator<String> i = collapse(li.iterator());
        while (i.hasNext()) {
            System.out.println(i.next());
        }
    }
}


```

```

1426 class foo {
1427     public static void aaarg() { System.out.println("aaarg"); }
1428 }
1429
1430 public class beta {
1431     static Random rand = new Random();
1432     private static final double gam(int x) {
1433         double result = 0;
1434         for (int i = 1; i <= x; i++) {
1435             result += Math.log(rand.nextDouble());
1436         }
1437         return result;
1438     }
1439     public static double draw(int a, int b) {
1440         return gam(a) / (gam(a) + gam(b));
1441     }
1442     public static void main(String[] args) {
1443         for (int i = 0; i < 100000; i++) {
1444             System.out.println(draw(2, 3));
1445         }
1446     }
1447 }
1448
1449 class Thing {
1450     public Thing() {
1451         Runtime.getRuntime().addShutdownHook(new Thread() {
1452             public void run() { flush(); }
1453         });
1454     }
1455     public void flush() { /* do some deferred action */ }
1456     public static void main(String[] args) {
1457         Thing t;
1458         while (true) { t = new Thing(); }
1459     }
1460 }
1461
1462 class crash {
1463     public void main(String[] args) {
1464         Object[] o = new Object[1];
1465         while (null != o) { o = new Object[1](o); }
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             public void remove() {
1475                 throw new UnsupportedOperationException("Not supported");
1476             }
1477             public T next() {
1478                 if (hasNext()) {
1479                     throw new NoSuchElementException("No more elements");
1480                 }
1481                 return buffer.next();
1482             }
1483             public boolean hasNext() {
1484                 while (source.hasNext()
1485                     && (null == buffer
1486                       || !buffer.hasNext())) {
1487

```

```

1488         }
1489         return buffer.hasNext();
1490     }
1491 }
1492
1493 public static void main(String[] args) {
1494     List<String> list = new ArrayList<String>();
1495     List<String> list2 = new ArrayList<String>();
1496     li.add("foo"); li.add("bar");
1497     li.add(list.iterator());
1498     l = new ArrayList<String>();
1499     l.add("baz"); l.add("quax");
1500     li.add(list.iterator());
1501     Iterator<String> i = collapse(li.iterator());
1502     while (i.hasNext()) {
1503         System.out.println(i.next());
1504     }
1505 }
1506
1507 class foo {
1508     public static void aaarg() { System.out.println("aaarg"); }
1509 }
1510
1511 public class beta {
1512     static Random rand = new Random();
1513     private static final double gam(int x) {
1514         double result = 0;
1515         for (int i = 1; i <= x; i++) {
1516             result += Math.log(rand.nextDouble());
1517         }
1518         return result;
1519     }
1520     public static double draw(int a, int b) {
1521         return gam(a) / (gam(a) + gam(b));
1522     }
1523     public static void main(String[] args) {
1524         for (int i = 0; i < 100000; i++) {
1525             System.out.println(draw(2, 5));
1526         }
1527     }
1528 }
1529
1530 class Thing {
1531     public Thing() {
1532         Runtime.getRuntime().addShutdownHook(new Thread() {
1533             public void run() { flush(); }
1534         });
1535     }
1536     public void flush() { /* do some deferred action */ }
1537     public static void main(String[] args) {
1538         Thing t;
1539         while (true) { t = new Thing(); }
1540     }
1541 }
1542
1543 class crash {
1544     public boolean hasNext() {
1545         while (source.hasNext()
1546             && (null != o) { o = new Object[1](o); }
1547

```

```
1596     }
1597 }
1598
1599 class GenericDemo {
1600     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
1601 ) {
1602     return new Iterator<T>() {
1603         private Iterator<? extends T> buffer = null;
1604         public void remove() {
1605             throw new UnsupportedOperationException("Not supported");
1606         }
1607         public T next() {
1608             if (i.hasNext()) {
1609                 throw new NoSuchElementException("No more elements");
1610             }
1611             return buffer.next();
1612         }
1613         public boolean hasNext() {
1614             while (source.hasNext())
1615                 && (null == buffer
1616                    || !buffer.hasNext());
1617             buffer = source.next();
1618         }
1619         return buffer.hasNext();
1620     };
1621 }
1622
1623 public static void main(String[] args) {
1624     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1625     li.add("foo"); li.add("bar");
1626     li.add(li.iterator());
1627     li = new ArrayList<String>();
1628     li.add("baz"); li.add("quux");
1629     li.add(li.iterator());
1630     Iterator<String> i = collapse(li.iterator());
1631     while (i.hasNext()) {
1632         System.out.println(i.next());
1633     }
1634 }
1635
1636 class foo {
1637     public static void aaarg() { System.out.println("aaarg"); }
1638 }
1639
1640 public class beta {
1641     static Random rand = new Random();
1642     private static final double gam(int x) {
1643         double result = 0;
1644         for (int i = 1; i <= x; i++) {
1645             result += Math.log(rand.nextDouble());
1646         }
1647         return result;
1648     }
1649     public static double draw(int a, int b) {
1650         return gam(a) / (gam(a) + gam(b));
1651     }
1652     public static void main(String[] args) {
1653         for (int i = 0; i < 100000; i++) {
```

```
1654     }
1655     System.out.println(draw(2, 5));
1656 }
1657
1658 class Thing {
1659     public Thing() {
1660         Runtime.getRuntime().addShutdownHook(new Thread() {
1661             public void run() { flush(); }
1662         });
1663     }
1664     public void flush() { /* do some deferred action */ }
1665     public static void main(String[] args) {
1666         Thing t;
1667         while (true) { t = new Thing(); }
1668     }
1669 }
1670
1671 class crash {
1672     public static void main(String[] args) {
1673         Object[] o = new Object[]();
1674         while (null != o) { o = new Object[]{}; }
1675     }
1676 }
1677
1678 class GenericDemo {
1679     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
1680 ) {
1681     return new Iterator<T>() {
1682         private Iterator<? extends T> buffer = null;
1683         public void remove() {
1684             throw new UnsupportedOperationException("Not supported");
1685         }
1686         public T next() {
1687             if (i.hasNext()) {
1688                 throw new NoSuchElementException("No more elements");
1689             }
1690             return buffer.next();
1691         }
1692         public boolean hasNext() {
1693             while (source.hasNext()
1694                    && (null == buffer
1695                       || !buffer.hasNext())) {
1696                 buffer = source.next();
1697             }
1698             return buffer.hasNext();
1699         }
1700     };
1701 }
1702
1703 public static void main(String[] args) {
1704     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1705     List<String> l = new ArrayList<String>();
1706     li.add("foo"); li.add("bar");
1707     li.add(li.iterator());
1708     li = new ArrayList<String>();
1709     li.add("baz"); li.add("quux");
1710     li.add(li.iterator());
1711     Iterator<String> i = collapse(li.iterator());
1712     while (i.hasNext()) {
1713         System.out.println(i.next());
1714     }
1715 }
```

```
1711 class foo {  
1712     public static void aaarg() { System.out.println("aaarg"); }  
1713 }  
1714  
1715 public class beta {  
1716     static Random rand = new Random();  
1717     private static final double gam(int x) {  
1718         double result = 0;  
1719         for (int i = 1; i <= x; i++) {  
1720             result += Math.log(rand.nextDouble());  
1721         }  
1722         return result;  
1723     }  
1724     public static double draw(int a, int b) {  
1725         return gam(a) / (gam(a) + gam(b));  
1726     }  
1727     public static void main(String[] args) {  
1728         for (int i = 0; i < 100000; i++) {  
1729             System.out.println(draw(2, 3));  
1730         }  
1731     }  
1732 }  
1733  
1734 class Thing {  
1735     public Thing() {  
1736         Runtime.getRuntime().addShutdownHook(new Thread() {  
1737             public void run() { flush(); }  
1738         });  
1739     }  
1740     public void flush() { /* do some deferred action */ }  
1741     public static void main(String[] args) {  
1742         Thing t;  
1743         while (true) { t = new Thing(); }  
1744     }  
1745 }  
1746  
1747 class crash {  
1748     public static void main(String[] args) {  
1749         Object[] o = new Object[1];  
1750         while (null != o) { o = new Object[1](o); }  
1751     }  
1752 }  
1753  
1754 class GenericDemo {  
1755     public static <T> Iterator<T> collapse(final Iterator<? extends T> source  
1756     ) {  
1757         return new Iterator<T>() {  
1758             private Iterator<? extends T> buffer = null;  
1759             public void remove() {  
1760                 throw new UnsupportedOperationException("Not supported");  
1761             }  
1762             public T next() {  
1763                 if (buffer == null) {  
1764                     throw new NoSuchElementException("No more elements");  
1765                 }  
1766                 return buffer.next();  
1767             }  
1768             public boolean hasNext() {  
1769                 while (source.hasNext())  
1770                     && (null == buffer  
1771                     || !buffer.hasNext()); }  
1772 }  
1773 }  
1774 }  
1775 }  
1776 }  
1777 }  
1778 }  
1779 }  
1780 }
```

```
1781     }  
1782     buffer = source.next();  
1783 }  
1784     return buffer.hasNext();  
1785 }  
1786 }  
1787 }  
1788 }  
1789 }  
1790 }  
1791 }  
1792 }  
1793 }  
1794 }  
1795 }  
1796 }  
1797 }  
1798 }  
1799 }  
1800 }  
1801 }  
1802 }  
1803 }  
1804 }  
1805 }  
1806 }  
1807 }  
1808 }  
1809 }  
1810 }  
1811 }  
1812 }  
1813 }  
1814 }  
1815 }  
1816 }  
1817 }  
1818 }  
1819 }  
1820 }  
1821 }  
1822 }  
1823 }  
1824 }  
1825 }  
1826 }  
1827 }  
1828 }  
1829 }  
1830 }  
1831 }  
1832 }  
1833 }  
1834 }  
1835 }  
1836 }  
1837 }  
1838 }  
1839 }  
1840 }  
1841 }  
1842 }  
1843 }  
1844 }  
1845 }  
1846 }  
1847 }  
1848 }  
1849 }  
1850 }  
1851 }  
1852 }  
1853 }
```

```
1854     }
1855     }
1856     }
1857     }
1858     }
1859     }
1860     }
1861     }
1862     }
1863     }
1864     }
1865     }
1866     }
1867     }
1868     }
1869     }
1870     }
1871     }
1872     }
1873     }
1874     }
1875     }
1876     }
1877     }
1878     }
1879     }
1880     }
1881     }
1882     }
1883     }
1884     }
1885     }
1886     }
1887     }
1888     }
1889     }
1890     }
1891     }
1892     }
1893     }
1894     }
1895     }
1896     }
1897     }
1898     }
1899     }
1900     }
1901     }
1902     }
1903     }
1904     }
1905     }
1906     }
1907     }
1908     }
1909     }
1910     }
1911     }
1912     }
1913     }
1914     }
1915     }
1916     }
1917     }
1918     }
1919     }
1920     }
1921     }
1922     }
1923     }
1924     }

class GenericDemo {
    private Iterator<T> extends T> buffer = null;

    public void remove() {
        throw new UnsupportedOperationException("Not supported");
    }

    public T next() {
        if (!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> l1 = new ArrayList<String>();
        l1.add("foo"); l1.add("bar");
        l1.add(l1.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        l1.add(l.iterator());

        Iterator<String> i = collapse(l1.iterator());
        while (i.hasNext()) {
            System.out.println(i.next());
        }
    }

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

    public class beta {
        static Random rand = new Random();
        private static final double gam(int x) {
            double result = 1.0;
            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) {
```

```
1925     for (int i = 0; i < 100000; i++) {
1926         System.out.println(draw(2, 5));
1927     }
1928     }
1929     }
1930     }
1931     }
1932     }
1933     }
1934     }
1935     }
1936     }
1937     }
1938     }
1939     }
1940     }
1941     }
1942     }
1943     }
1944     }
1945     }
1946     }
1947     }
1948     }
1949     }
1950     }
1951     }
1952     }
1953     }
1954     }
1955     }
1956     }
1957     }
1958     }
1959     }
1960     }
1961     }
1962     }
1963     }
1964     }
1965     }
1966     }
1967     }
1968     }
1969     }
1970     }
1971     }
1972     }
1973     }
1974     }
1975     }
1976     }
1977     }
1978     }
1979     }
1980     }
1981     }
1982     }
1983     }
1984     }
1985     }
1986     }
1987     }
1988     }
1989     }
1990     }
1991     }
1992     }
1993     }
1994     }
1995     }

for (int i = 0; i < 100000; i++) {
    System.out.println(draw(2, 5));
}

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[0]; }
    }
}

class GenericDemo {
    public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<T> source
    ) {
        return new Iterator<T>() {
            private Iterator<T> extends T> buffer = null;

            public void remove() {
                throw new UnsupportedOperationException("Not supported");
            }

            public T next() {
                if (!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> l1 = new ArrayList<String>();
                l1.add("foo"); l1.add("bar");
                l1.add(l1.iterator());

                l = new ArrayList<String>();
                l.add("baz"); l.add("quux");
                l1.add(l.iterator());

                Iterator<String> i = collapse(l1.iterator());
                while (i.hasNext()) {
                    System.out.println(i.next());
                }
            }

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

            public class beta {
                static Random rand = new Random();
                private static final double gam(int x) {
                    double result = 1.0;
                    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) {
```

```
1996 }
1997
1998
1999
2000 class foo {
2001     public static void aaarg() { System.out.println("aaarg"); }
2002 }
2003
2004 public class beta {
2005     static Random rand = new Random();
2006     private static final double gam(int x) {
2007         double result = 0;
2008         for (int i = 1; i <= x; i++) {
2009             result += Math.log(rand.nextDouble());
2010         }
2011         return result;
2012     }
2013     public static double draw(int a, int b) {
2014         return gam(a) / (gam(a) + gam(b));
2015     }
2016     public static void main(String[] args) {
2017         for (int i = 0; i < 100000; i++) {
2018             System.out.println(draw(2, 5));
2019         }
2020     }
2021 }
2022
2023
2024
2025
2026 class Thing {
2027     public Thing() {
2028         Runtime.getRuntime().addShutdownHook(new Thread() {
2029             public void run() { flush(); }
2030         });
2031     }
2032     public void flush() { /* do some deferred action */ }
2033     public static void main(String[] args) {
2034         Thing t;
2035         while (true) { t = new Thing(); }
2036     }
2037 }
2038
2039
2040 class GenericDemo {
2041     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
2042         return new Iterator<T>() {
2043             private Iterator<? extends T> buffer = null;
2044             public void remove() {
2045                 throw new UnsupportedOperationException("Not supported");
2046             }
2047             public T next() {
2048                 if (buffer == null) {
2049                     throw new NoSuchElementException("No more elements");
2050                 }
2051                 return buffer.next();
2052             }
2053             public boolean hasNext() {
2054                 while (source.hasNext()
2055                     && (null == buffer
2056
```

```
2057         || !buffer.hasNext()) {
2058             buffer = source.next();
2059         }
2060         return buffer.hasNext();
2061     }
2062 }
2063
2064 public static void main(String[] args) {
2065     List<String> l = new ArrayList<String>();
2066     List<String> l2 = new ArrayList<String>();
2067     l.add("foo"); l.add("bar");
2068     l2.add(l.iterator());
2069     l = new ArrayList<String>();
2070     l.add("baz"); l.add("quux");
2071     l2.add(l.iterator());
2072 }
2073
2074 Iterator<String> i = collapse(l2.iterator());
2075 while (i.hasNext()) {
2076     System.out.println(i.next());
2077 }
2078
2079
2080
2081 class foo {
2082     public static void aaarg() { System.out.println("aaarg"); }
2083 }
2084
2085 public class beta {
2086     static Random rand = new Random();
2087     private static final double gam(int x) {
2088         double result = 0;
2089         for (int i = 1; i <= x; i++) {
2090             result += Math.log(rand.nextDouble());
2091         }
2092         return result;
2093     }
2094     public static double draw(int a, int b) {
2095         return gam(a) / (gam(a) + gam(b));
2096     }
2097     public static void main(String[] args) {
2098         for (int i = 0; i < 100000; i++) {
2099             System.out.println(draw(2, 5));
2100         }
2101     }
2102 }
2103
2104
2105 class crash {
2106     public static void main(String[] args) {
2107         Object[] o = new Object[10];
2108         while (null != o) { o = new Object[10]; }
2109     }
2110 }
2111
2112
2113 class GenericDemo {
2114     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
2115         return new Iterator<T>() {
2116             private Iterator<? extends T> buffer = null;
2117             public void remove() {
2118                 throw new UnsupportedOperationException("Not supported");
2119             }
2120             public T next() {
2121                 if (buffer == null) {
2122                     throw new NoSuchElementException("No more elements");
2123                 }
2124                 return buffer.next();
2125             }
2126             public boolean hasNext() {
2127                 while (source.hasNext()
2128                     && (null == buffer
2129
```

```
2159     } while (null != o) { o = new Object[1][0]; }
2160     }
2161 }
2162
2163 class GenericDemo {
2164     public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<? extends T>> source
2165 ) {
2166     return new Iterator<T>() {
2167         private Iterator<T> extends T; buffer = null;
2168         public void remove() {
2169             throw new UnsupportedOperationException("Not supported");
2170         }
2171         public T next() {
2172             if (!hasNext()) {
2173                 throw new NoSuchElementException("No more elements");
2174             }
2175             return buffer.next();
2176         }
2177         public boolean hasNext() {
2178             while (source.hasNext())
2179                 && (null == buffer
2180                    || !buffer.hasNext()) {
2181                 buffer = source.next();
2182             }
2183             return buffer.hasNext();
2184         }
2185     };
2186 }
2187
2188     public static void main(String[] args) {
2189         list<Iterator<String>> li = new ArrayList<Iterator<String>>();
2190         list<String> l = new ArrayList<String>();
2191         l.add("foo"); l.add("bar");
2192         li.add(l.iterator());
2193         l = new ArrayList<String>();
2194         l.add("baz"); l.add("qux");
2195         li.add(l.iterator());
2196         Iterator<String> i = collapse(li.iterator());
2197         while (i.hasNext()) {
2198             System.out.println(i.next());
2199         }
2200     }
2201 }
2202
2203 class foo {
2204     public static void aaarg() { System.out.println("aaarg"); }
2205 }
2206
2207 public class beta {
2208     static Random rand = new Random();
2209     private static final double gam(int x) {
2210         double val = 0;
2211         for (int i = 1; i <= x; i++) {
2212             result += Math.Log(rand.nextDouble());
2213         }
2214         return result;
2215     }
2216     public static double draw(int a, int b) {
2217         return gam(a) / (gam(a) + gam(b));
2218     }
2219     public static void main(String[] args) {
```

```
2220         for (int i = 0; i < 100000; i++) {
2221             System.out.println(draw(2, 5));
2222         }
2223     }
2224 }
2225
2226 class Thing {
2227     public Thing() {
2228         Runtime.getRuntime().addShutdownHook(new Thread() {
2229             public void run() { flush(); }
2230         });
2231     }
2232     public void flush() { /* do some deferred action */ }
2233     public static void main(String[] args) {
2234         Thing t;
2235         while (true) { t = new Thing(); }
2236     }
2237 }
2238
2239     class crash {
2240     public static void main(String[] args) {
2241         Object[] o = new Object[1];
2242         while (null != o) { o = new Object[1][0]; }
2243     }
2244 }
2245
2246     class GenericDemo {
2247     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
2248 ) {
2249         return new Iterator<T>() {
2250             private Iterator<? extends T> buffer = null;
2251             public void remove() {
2252                 throw new UnsupportedOperationException("Not supported");
2253             }
2254             public T next() {
2255                 if (!hasNext()) {
2256                     throw new NoSuchElementException("No more elements");
2257                 }
2258                 return buffer.next();
2259             }
2260             public boolean hasNext() {
2261                 while (source.hasNext()
2262                    && (null == buffer
2263                       || !buffer.hasNext())) {
2264                     buffer = source.next();
2265                 }
2266                 return buffer.hasNext();
2267             }
2268         };
2269     }
2270     public static void main(String[] args) {
2271         list<Iterator<String>> li = new ArrayList<Iterator<String>>();
2272         list<String> l = new ArrayList<String>();
2273         l.add("foo"); l.add("bar");
2274         li.add(l.iterator());
2275         l = new ArrayList<String>();
2276         l.add("baz"); l.add("qux");
2277         li.add(l.iterator());
2278         Iterator<String> i = collapse(li.iterator());
2279         while (i.hasNext()) {
2280             System.out.println(i.next());
2281         }
2282     }
2283 }
```



```

2287 }
2288 }
2289 }
2290 }
2291 }
2292 }
2293 }
2294 }
2295 }
2296 }
2297 }
2298 }
2299 }
2300 }
2301 }
2302 }
2303 }
2304 }
2305 }
2306 }
2307 }
2308 }
2309 }
2310 }
2311 }
2312 }
2313 }
2314 }
2315 }
2316 }
2317 }
2318 }
2319 }
2320 }
2321 }
2322 }
2323 }
2324 }
2325 }
2326 }
2327 }
2328 }
2329 }
2330 }
2331 }
2332 }
2333 }
2334 }
2335 }
2336 }
2337 }
2338 }
2339 }
2340 }
2341 }
2342 }
2343 }
2344 }
2345 }
2346 }
2347 }
2348 }
2349 }
2350 }
2351 }
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 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 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 (!hasNext()) {
                    throw new NoSuchElementException("No more elements");
                }
                return buffer.next();
            }
            public boolean hasNext() {
                while (source.hasNext())
                    && (null != buffer

```

```

2352 }
2353 }
2354 }
2355 }
2356 }
2357 }
2358 }
2359 }
2360 }
2361 }
2362 }
2363 }
2364 }
2365 }
2366 }
2367 }
2368 }
2369 }
2370 }
2371 }
2372 }
2373 }
2374 }
2375 }
2376 }
2377 }
2378 }
2379 }
2380 }
2381 }
2382 }
2383 }
2384 }
2385 }
2386 }
2387 }
2388 }
2389 }
2390 }
2391 }
2392 }
2393 }
2394 }
2395 }
2396 }
2397 }
2398 }
2399 }
2400 }
2401 }
2402 }
2403 }
2404 }
2405 }
2406 }
2407 }
2408 }
2409 }
2410 }
2411 }
2412 }
2413 }
2414 }
2415 }
2416 }
2417 }
2418 }
2419 }
2420 }
2421 }
2422 }
2423 }
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 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[10];
        while (null != o) { o = new Object[10]; }
    }
}

```

```
2424     } while (null != o) { o = new Object[1][0]; }
2425     }
2426 }
2427
2428 class GenericDemo {
2429     public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<? extends T>> source
2430 ) {
2431     return new Iterator<T>() {
2432         private Iterator<T> 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                 && (null != buffer
2445                     || !buffer.hasNext());
2446                 buffer = source.next();
2447             }
2448             return buffer.hasNext();
2449         }
2450     };
2451 }
2452
2453     public static void main(String[] args) {
2454         list<Iterator<String>> li = new ArrayList<Iterator<String>>();
2455         list<String> l = new ArrayList<String>();
2456         l.add("foo"); l.add("bar");
2457         li.add(li.iterator());
2458         l = new ArrayList<String>();
2459         l.add("baz"); l.add("quux");
2460         li.add(li.iterator());
2461         Iterator<String> i = collapse(li.iterator());
2462         while (i.hasNext()) {
2463             System.out.println(i.next());
2464         }
2465     }
2466 }
2467
2468 class foo {
2469     public static void aaarg() { System.out.println("aaarg"); }
2470 }
2471
2472 public class beta {
2473     static Random rand = new Random();
2474     private static final double gam(int x) {
2475         double val = 0;
2476         for (int i = 1; i <= x; i++) {
2477             result += Math.Log(rand.nextDouble());
2478         }
2479         return result;
2480     }
2481     public static double draw(int a, int b) {
2482         return gam(a) / (gam(a) + gam(b));
2483     }
2484     public static void main(String[] args) {
2485 
```

```
2486         for (int i = 0; i < 100000; i++) {
2487             System.out.println(draw(2, 5));
2488         }
2489     }
2490 }
2491
2492 class Thing {
2493     public Thing() {
2494         Runtime.getRuntime().addShutdownHook(new Thread() {
2495             public void run() { flush(); }
2496         });
2497     }
2498     public void flush() { /* do some deferred action */ }
2499     public static void main(String[] args) {
2500         Thing t;
2501         while (true) { t = new Thing(); }
2502     }
2503 }
2504
2505 class crash {
2506     public static void main(String[] args) {
2507         Object[] o = new Object[1];
2508         while (null != o) { o = new Object[1][0]; }
2509     }
2510 }
2511
2512 class GenericDemo {
2513     public static <T> Iterator<T> collapse(final Iterator<? extends T>> extends T>> source
2514 ) {
2515     return new Iterator<T>() {
2516         private Iterator<T> extends T; buffer = null;
2517         public void remove() {
2518             throw new UnsupportedOperationException("Not supported");
2519         }
2520         public T next() {
2521             if (!hasNext()) {
2522                 throw new NoSuchElementException("No more elements");
2523             }
2524             return buffer.next();
2525         }
2526         public boolean hasNext() {
2527             while (source.hasNext()
2528                 && (null != buffer
2529                     || !buffer.hasNext())) {
2530                 buffer = source.next();
2531             }
2532             return buffer.hasNext();
2533         }
2534     };
2535 }
2536
2537     public static void main(String[] args) {
2538         list<Iterator<String>> li = new ArrayList<Iterator<String>>();
2539         list<String> l = new ArrayList<String>();
2540         l.add("foo"); l.add("bar");
2541         li.add(li.iterator());
2542         l = new ArrayList<String>();
2543         l.add("baz"); l.add("quux");
2544         li.add(li.iterator());
2545         Iterator<String> i = collapse(li.iterator());
2546         while (i.hasNext()) {
2547             System.out.println(i.next());
2548         }
2549     }
2550 }
2551
2552 public static void main(String[] args) {
2553     list<Iterator<String>> li = new ArrayList<Iterator<String>>();
2554     list<String> l = new ArrayList<String>();
2555     l.add("foo"); l.add("bar");
2556     li.add(li.iterator());
2557     l = new ArrayList<String>();
2558     l.add("baz"); l.add("quux");
2559     li.add(li.iterator());
2560     Iterator<String> i = collapse(li.iterator());
2561     while (i.hasNext()) {
2562         System.out.println(i.next());
2563     }
2564 }
2565 }
```

```

2566 }
2567 }
2568 class foo {
2569     public static void aaarg() { System.out.println("aaarg"); }
2570 }
2571 }
2572 public class beta {
2573     static Random rand = new Random();
2574     private static final double gam(int x) {
2575         double result = 0;
2576         for (int i = 1; i <= x; i++) {
2577             result += Math.log(rand.nextDouble());
2578         }
2579         return result;
2580     }
2581     public static double draw(int a, int b) {
2582         return gam(a) / (gam(a) + gam(b));
2583     }
2584     public static void main(String[] args) {
2585         for (int i = 0; i < 100000; i++) {
2586             System.out.println(draw(2, 5));
2587         }
2588     }
2589 }
2590 }
2591 class Thing {
2592     public Thing() {
2593         Runtime.getRuntime().addShutdownHook(new Thread() {
2594             public void run() { flush(); }
2595         });
2596     }
2597     public void flush() { /* do some deferred action */ }
2598     public static void main(String[] args) {
2599         Thing t;
2600         while (true) { t = new Thing(); }
2601     }
2602 }
2603 }
2604 class GenericDemo {
2605     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source) {
2606         return new Iterator<T>() {
2607             private Iterator<? extends T> buffer = null;
2608             public void remove() {
2609                 throw new UnsupportedOperationException("Not supported");
2610             }
2611             public T next() {
2612                 if (!hasNext()) {
2613                     throw new NoSuchElementException("No more elements");
2614                 }
2615                 return buffer.next();
2616             }
2617             public boolean hasNext() {
2618                 while (source.hasNext())
2619                     && (null == buffer

```

```

2619         }
2620     }
2621     public void flush() { /* do some deferred action */ }
2622     public static void main(String[] args) {
2623         Thing t;
2624         while (true) { t = new Thing(); }
2625     }
2626 }
2627 class crash {
2628     public static void main(String[] args) {
2629         Object[] o = new Object[1];
2630     }
2631 }
2632 }
2633 class foo {
2634     public static void aaarg() { System.out.println("aaarg"); }
2635 }
2636 }
2637 public class beta {
2638     static Random rand = new Random();
2639     private static final double gam(int x) {
2640         double result = 0;
2641         for (int i = 1; i <= x; i++) {
2642             result += Math.log(rand.nextDouble());
2643         }
2644         return result;
2645     }
2646     public static double draw(int a, int b) {
2647         return gam(a) / (gam(a) + gam(b));
2648     }
2649     public static void main(String[] args) {
2650         for (int i = 0; i < 100000; i++) {
2651             System.out.println(draw(2, 5));
2652         }
2653     }
2654 }
2655 }
2656 class Thing {
2657     public Thing() {
2658         Runtime.getRuntime().addShutdownHook(new Thread() {
2659             public void run() { flush(); }
2660         });
2661     }
2662     public void flush() { /* do some deferred action */ }
2663     public static void main(String[] args) {
2664         Thing t;
2665         while (true) { t = new Thing(); }
2666     }
2667 }
2668 }
2669 class crash {
2670     public static void main(String[] args) {
2671         Object[] o = new Object[1];
2672     }
2673 }

```

```

2700     while (null != o) { o = new Object[]{}; }
2701     }
2702 }
2703
2704 class GenericDemo {
2705     public static <T> Iterator<T> collapse(final Iterator<T> extends Iterator<? extends T>> source
2706 ) {
2707     return new Iterator<T>() {
2708         private Iterator<T> extends T> buffer = null;
2709         public void remove() {
2710             throw new UnsupportedOperationException("Not supported");
2711         }
2712         public T next() {
2713             if (!hasNext()) {
2714                 throw new NoSuchElementException("No more elements");
2715             }
2716             return buffer.next();
2717         }
2718         public boolean hasNext() {
2719             while (source.hasNext()) {
2720                 && (null == buffer
2721                 || !buffer.hasNext()) {
2722                     buffer = source.next();
2723                 }
2724                 return buffer.hasNext();
2725             }
2726         }
2727     };
2728 }
2729
2730 public static void main(String[] args) {
2731     List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2732     List<String> l = new ArrayList<String>();
2733     l.add("foo"); l.add("bar");
2734     li.add(l.iterator());
2735     l = new ArrayList<String>();
2736     l.add("baz"); l.add("qux");
2737     li.add(l.iterator());
2738     Iterator<String> i = collapse(li.iterator());
2739     while (i.hasNext()) {
2740         System.out.println(i.next());
2741     }
2742 }
2743
2744 }
2745
2746 }
2747
2748 }
2749
2750 }
2751
2752 }
2753
2754 }
2755
2756 }
2757
2758 }
2759
2760 }
2761
2762 }
2763
2764 }
2765
2766 }
2767
2768 }
2769
2770 }
2771
2772 }
2773
2774 }
2775
2776 }
2777
2778 }
2779
2780 }
2781
2782 }
2783
2784 }
2785
2786 }
2787
2788 }
2789
2790 }
2791
2792 }
2793
2794 }
2795
2796 }
2797
2798 }
2799
2800 }
2801
2802 }
2803
2804 }
2805
2806 }
2807
2808 }
2809
2810 }
2811
2812 }
2813
2814 }
2815
2816 }
2817
2818 }
2819
2820 }
2821
2822 }
2823
2824 }
2825
2826 }
2827
2828 }
2829
2830 }
2831
2832 }
2833
2834 }
2835
2836 }
2837
2838 }
2839
2840 }
2841
2842 }
2843
2844 }
2845
2846 }
2847
2848 }
2849
2850 }
2851
2852 }
2853
2854 }
2855
2856 }
2857
2858 }
2859
2860 }
2861
2862 }
2863
2864 }
2865
2866 }
2867
2868 }
2869
2870 }
2871
2872 }
2873
2874 }
2875
2876 }
2877
2878 }
2879
2880 }
2881
2882 }
2883
2884 }
2885
2886 }
2887
2888 }
2889
2890 }
2891
2892 }
2893
2894 }
2895
2896 }
2897
2898 }
2899
2900 }
2901
2902 }
2903
2904 }
2905
2906 }
2907
2908 }
2909
2910 }
2911
2912 }
2913
2914 }
2915
2916 }
2917
2918 }
2919
2920 }
2921
2922 }
2923
2924 }
2925
2926 }
2927
2928 }
2929
2930 }
2931
2932 }
2933
2934 }
2935
2936 }
2937
2938 }
2939
2940 }
2941
2942 }
2943
2944 }
2945
2946 }
2947
2948 }
2949
2950 }
2951
2952 }
2953
2954 }
2955
2956 }
2957
2958 }
2959
2960 }
2961
2962 }
2963
2964 }
2965
2966 }
2967
2968 }
2969
2970 }
2971
2972 }
2973
2974 }
2975
2976 }
2977
2978 }
2979
2980 }
2981
2982 }
2983
2984 }
2985
2986 }
2987
2988 }
2989
2990 }
2991
2992 }
2993
2994 }
2995
2996 }
2997
2998 }
2999
3000 }

```