```
import java.util.Random;

import java.util.*;

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[]{};
        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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{};
```

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



```
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
513     public void flush() { /* do some deferred action */ }
514
515     public static void main(String[] args) {
516         Thing t;
517         while (true) { t = new Thing(); }
518     }
519 }
520
521 class crash {
522     public static void main(String[] args) {
523         Object[] o = new Object[]{};
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         return new Iterator<T>() {
531             private Iterator<? extends T> buffer = null;
532
533             public void remove() {
534                 throw new UnsupportedOperationException("Not supported");
535             }
536
537             public T next() {
538                 if (!hasNext()) {
539                     throw new NoSuchElementException("No more elements");
540                 }
541                 return buffer.next();
542             }
543
544             public boolean hasNext() {
545                 while (source.hasNext()
546                         && (null == buffer
547                             || !buffer.hasNext())) {
548                     buffer = source.next();
549                 }
550
551                 return buffer.hasNext();
552             }
553         };
554     }
555
556     public static void main(String[] args) {
557         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
558         List<String> l = new ArrayList<String>();
559         l.add("foo"); l.add("bar");
560         li.add(l.iterator());
561
562         l = new ArrayList<String>();
563         l.add("baz"); l.add("quux");
564         li.add(l.iterator());
565
566         Iterator<String> i = collapse(li.iterator());
567         while (i.hasNext()) {
568             System.out.println(i.next());
569         }
570     }
```

```
}


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[]{};
        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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

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

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

```
856
857
858 class foo {
859     public static void aaarg() { System.out.println("aaarg"); }
860 }
861
862 public class beta {
863
864     static Random rand = new Random();
865
866     private static final double gam(int x) {
867         double result = 0;
868         for (int i = 1; i <= x; i++) {
869             result += Math.log(rand.nextDouble());
870         }
871         return result;
872     }
873
874     public static double draw(int a, int b) {
875         return gam(a) / (gam(a) + gam(b));
876     }
877
878     public static void main(String[] args) {
879         for (int i = 0; i < 100000; i++) {
880             System.out.println(draw(2, 5));
881         }
882     }
883 }
884
885 class Thing {
886     public Thing() {
887         Runtime.getRuntime().addShutdownHook(new Thread() {
888                 public void run() { flush(); }
889             });
890     }
891
892     public void flush() { /* do some deferred action */ }
893
894     public static void main(String[] args) {
895         Thing t;
896         while (true) { t = new Thing(); }
897     }
898 }
899
900 class crash {
901     public static void main(String[] args) {
902         Object[] o = new Object[]{};
903         while (null != o) { o = new Object[]{o}; }
904     }
905 }
906
907 class GenericDemo {
908     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
    ) {
909         return new Iterator<T>() {
910             private Iterator<? extends T> buffer = null;
911
912             public void remove() {
913                 throw new UnsupportedOperationException("Not supported");
914             }
915
916             public T next() {
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()
925                        && (null == buffer
926                            || !buffer.hasNext())) {
```



```
927                     buffer = source.next();
928                 }
929
930                 return buffer.hasNext();
931             }
932         };
933     }
934
935     public static void main(String[] args) {
936         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
937         List<String> l = new ArrayList<String>();
938         l.add("foo"); l.add("bar");
939         li.add(l.iterator());
940
941         l = new ArrayList<String>();
942         l.add("baz"); l.add("quux");
943         li.add(l.iterator());
944
945         Iterator<String> i = collapse(li.iterator());
946         while (i.hasNext()) {
947             System.out.println(i.next());
948         }
949     }
950 }
951
952
953 class foo {
954     public static void aaarg() { System.out.println("aaarg"); }
955 }
956
957 public class beta {
958
959     static Random rand = new Random();
960
961     private static final double gam(int x) {
962         double result = 0;
963         for (int i = 1; i <= x; i++) {
964             result += Math.log(rand.nextDouble());
965         }
966         return result;
967     }
968
969     public static double draw(int a, int b) {
970         return gam(a) / (gam(a) + gam(b));
971     }
972
973     public static void main(String[] args) {
974         for (int i = 0; i < 100000; i++) {
975             System.out.println(draw(2, 5));
976         }
977     }
978 }
979
980 class Thing {
981     public Thing() {
982         Runtime.getRuntime().addShutdownHook(new Thread() {
983                 public void run() { flush(); }
984             });
985     }
986
987     public void flush() { /* do some deferred action */ }
988
989     public static void main(String[] args) {
990         Thing t;
991         while (true) { t = new Thing(); }
992     }
993 }
994
995 class crash {
996     public static void main(String[] args) {
997         Object[] o = new Object[]{};
998         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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

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

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

```
1141
1142
1143 class foo {
1144     public static void aaarg() { System.out.println("aaarg"); }
1145 }
1146
1147 public class beta {
1148
1149     static Random rand = new Random();
1150
1151     private static final double gam(int x) {
1152         double result = 0;
1153         for (int i = 1; i <= x; i++) {
1154             result += Math.log(rand.nextDouble());
1155         }
1156         return result;
1157     }
1158
1159     public static double draw(int a, int b) {
1160         return gam(a) / (gam(a) + gam(b));
1161     }
1162
1163     public static void main(String[] args) {
1164         for (int i = 0; i < 100000; i++) {
1165             System.out.println(draw(2, 5));
1166         }
1167     }
1168 }
1169
1170 class Thing {
1171     public Thing() {
1172         Runtime.getRuntime().addShutdownHook(new Thread() {
1173                 public void run() { flush(); }
1174             });
1175     }
1176
1177     public void flush() { /* do some deferred action */ }
1178
1179     public static void main(String[] args) {
1180         Thing t;
1181         while (true) { t = new Thing(); }
1182     }
1183 }
1184
1185 class crash {
1186     public static void main(String[] args) {
1187         Object[] o = new Object[]{};
1188         while (null != o) { o = new Object[]{o}; }
1189     }
1190 }
1191
1192 class GenericDemo {
1193     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
    ) {
1194         return new Iterator<T>() {
1195             private Iterator<? extends T> buffer = null;
1196
1197             public void remove() {
1198                 throw new UnsupportedOperationException("Not supported");
1199             }
1200
1201             public T next() {
1202                 if (!hasNext()) {
1203                     throw new NoSuchElementException("No more elements");
1204                 }
1205                 return buffer.next();
1206             }
1207
1208             public boolean hasNext() {
1209                 while (source.hasNext()
1210                        && (null == buffer
1211                            || !buffer.hasNext())) {
```



```
1212                     buffer = source.next();
1213                 }
1214
1215                 return buffer.hasNext();
1216             }
1217         };
1218     }
1219
1220     public static void main(String[] args) {
1221         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1222         List<String> l = new ArrayList<String>();
1223         l.add("foo"); l.add("bar");
1224         li.add(l.iterator());
1225
1226         l = new ArrayList<String>();
1227         l.add("baz"); l.add("quux");
1228         li.add(l.iterator());
1229
1230         Iterator<String> i = collapse(li.iterator());
1231         while (i.hasNext()) {
1232             System.out.println(i.next());
1233         }
1234     }
1235 }
1236
1237
1238 class foo {
1239     public static void aaarg() { System.out.println("aaarg"); }
1240 }
1241
1242 public class beta {
1243
1244     static Random rand = new Random();
1245
1246     private static final double gam(int x) {
1247         double result = 0;
1248         for (int i = 1; i <= x; i++) {
1249             result += Math.log(rand.nextDouble());
1250         }
1251         return result;
1252     }
1253
1254     public static double draw(int a, int b) {
1255         return gam(a) / (gam(a) + gam(b));
1256     }
1257
1258     public static void main(String[] args) {
1259         for (int i = 0; i < 100000; i++) {
1260             System.out.println(draw(2, 5));
1261         }
1262     }
1263 }
1264
1265 class Thing {
1266     public Thing() {
1267         Runtime.getRuntime().addShutdownHook(new Thread() {
1268                 public void run() { flush(); }
1269             });
1270     }
1271
1272     public void flush() { /* do some deferred action */ }
1273
1274     public static void main(String[] args) {
1275         Thing t;
1276         while (true) { t = new Thing(); }
1277     }
1278 }
1279
1280 class crash {
1281     public static void main(String[] args) {
1282         Object[] o = new Object[]{};
1283         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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

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

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

```
1426
1427
1428 class foo {
1429     public static void aaarg() { System.out.println("aaarg"); }
1430 }
1431
1432 public class beta {
1433
1434     static Random rand = new Random();
1435
1436     private static final double gam(int x) {
1437         double result = 0;
1438         for (int i = 1; i <= x; i++) {
1439             result += Math.log(rand.nextDouble());
1440         }
1441         return result;
1442     }
1443
1444     public static double draw(int a, int b) {
1445         return gam(a) / (gam(a) + gam(b));
1446     }
1447
1448     public static void main(String[] args) {
1449         for (int i = 0; i < 100000; i++) {
1450             System.out.println(draw(2, 5));
1451         }
1452     }
1453 }
1454
1455 class Thing {
1456     public Thing() {
1457         Runtime.getRuntime().addShutdownHook(new Thread() {
1458                 public void run() { flush(); }
1459             });
1460     }
1461
1462     public void flush() { /* do some deferred action */ }
1463
1464     public static void main(String[] args) {
1465         Thing t;
1466         while (true) { t = new Thing(); }
1467     }
1468 }
1469
1470 class crash {
1471     public static void main(String[] args) {
1472         Object[] o = new Object[]{};
1473         while (null != o) { o = new Object[]{o}; }
1474     }
1475 }
1476
1477 class GenericDemo {
1478     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
    ) {
1479         return new Iterator<T>() {
1480             private Iterator<? extends T> buffer = null;
1481
1482             public void remove() {
1483                 throw new UnsupportedOperationException("Not supported");
1484             }
1485
1486             public T next() {
1487                 if (!hasNext()) {
1488                     throw new NoSuchElementException("No more elements");
1489                 }
1490                 return buffer.next();
1491             }
1492
1493             public boolean hasNext() {
1494                 while (source.hasNext()
1495                        && (null == buffer
1496                            || !buffer.hasNext())) {
```



```
1497                     buffer = source.next();
1498                 }
1499
1500                 return buffer.hasNext();
1501             }
1502         };
1503     }
1504
1505     public static void main(String[] args) {
1506         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1507         List<String> l = new ArrayList<String>();
1508         l.add("foo"); l.add("bar");
1509         li.add(l.iterator());
1510
1511         l = new ArrayList<String>();
1512         l.add("baz"); l.add("quux");
1513         li.add(l.iterator());
1514
1515         Iterator<String> i = collapse(li.iterator());
1516         while (i.hasNext()) {
1517             System.out.println(i.next());
1518         }
1519     }
1520 }
1521
1522
1523 class foo {
1524     public static void aaarg() { System.out.println("aaarg"); }
1525 }
1526
1527 public class beta {
1528
1529     static Random rand = new Random();
1530
1531     private static final double gam(int x) {
1532         double result = 0;
1533         for (int i = 1; i <= x; i++) {
1534             result += Math.log(rand.nextDouble());
1535         }
1536         return result;
1537     }
1538
1539     public static double draw(int a, int b) {
1540         return gam(a) / (gam(a) + gam(b));
1541     }
1542
1543     public static void main(String[] args) {
1544         for (int i = 0; i < 100000; i++) {
1545             System.out.println(draw(2, 5));
1546         }
1547     }
1548 }
1549
1550 class Thing {
1551     public Thing() {
1552         Runtime.getRuntime().addShutdownHook(new Thread() {
1553                 public void run() { flush(); }
1554             });
1555     }
1556
1557     public void flush() { /* do some deferred action */ }
1558
1559     public static void main(String[] args) {
1560         Thing t;
1561         while (true) { t = new Thing(); }
1562     }
1563 }
1564
1565 class crash {
1566     public static void main(String[] args) {
1567         Object[] o = new Object[]{};
1568         while (null != o) { o = new Object[]{o}; }
```

```
1569      }
1570  }
1571
1572  class GenericDemo {
1573      public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
      ) {
1574          return new Iterator<T>() {
1575              private Iterator<? extends T> buffer = null;
1576
1577              public void remove() {
1578                  throw new UnsupportedOperationException("Not supported");
1579              }
1580
1581              public T next() {
1582                  if (!hasNext()) {
1583                      throw new NoSuchElementException("No more elements");
1584                  }
1585                  return buffer.next();
1586              }
1587
1588              public boolean hasNext() {
1589                  while (source.hasNext()
1590                         && (null == buffer
1591                             || !buffer.hasNext())) {
1592                      buffer = source.next();
1593                  }
1594
1595                  return buffer.hasNext();
1596              }
1597          };
1598      }
1599
1600      public static void main(String[] args) {
1601          List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1602          List<String> l = new ArrayList<String>();
1603          l.add("foo"); l.add("bar");
1604          li.add(l.iterator());
1605
1606          l = new ArrayList<String>();
1607          l.add("baz"); l.add("quux");
1608          li.add(l.iterator());
1609
1610          Iterator<String> i = collapse(li.iterator());
1611          while (i.hasNext()) {
1612              System.out.println(i.next());
1613          }
1614      }
1615  }
1616
1617
1618  class foo {
1619      public static void aaarg() { System.out.println("aaarg"); }
1620  }
1621
1622  public class beta {
1623
1624      static Random rand = new Random();
1625
1626      private static final double gam(int x) {
1627          double result = 0;
1628          for (int i = 1; i <= x; i++) {
1629              result += Math.log(rand.nextDouble());
1630          }
1631          return result;
1632      }
1633
1634      public static double draw(int a, int b) {
1635          return gam(a) / (gam(a) + gam(b));
1636      }
1637
1638      public static void main(String[] args) {
1639          for (int i = 0; i < 100000; i++) {
```



```
1640              System.out.println(draw(2, 5));
1641          }
1642      }
1643  }
1644
1645  class Thing {
1646      public Thing() {
1647          Runtime.getRuntime().addShutdownHook(new Thread() {
1648                  public void run() { flush(); }
1649              });
1650      }
1651
1652      public void flush() { /* do some deferred action */ }
1653
1654      public static void main(String[] args) {
1655          Thing t;
1656          while (true) { t = new Thing(); }
1657      }
1658  }
1659
1660  class crash {
1661      public static void main(String[] args) {
1662          Object[] o = new Object[]{};
1663          while (null != o) { o = new Object[]{o}; }
1664      }
1665  }
1666
1667  class GenericDemo {
1668      public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
      ) {
1669          return new Iterator<T>() {
1670              private Iterator<? extends T> buffer = null;
1671
1672              public void remove() {
1673                  throw new UnsupportedOperationException("Not supported");
1674              }
1675
1676              public T next() {
1677                  if (!hasNext()) {
1678                      throw new NoSuchElementException("No more elements");
1679                  }
1680                  return buffer.next();
1681              }
1682
1683              public boolean hasNext() {
1684                  while (source.hasNext()
1685                         && (null == buffer
1686                             || !buffer.hasNext())) {
1687                      buffer = source.next();
1688                  }
1689
1690                  return buffer.hasNext();
1691              }
1692          };
1693      }
1694
1695      public static void main(String[] args) {
1696          List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1697          List<String> l = new ArrayList<String>();
1698          l.add("foo"); l.add("bar");
1699          li.add(l.iterator());
1700
1701          l = new ArrayList<String>();
1702          l.add("baz"); l.add("quux");
1703          li.add(l.iterator());
1704
1705          Iterator<String> i = collapse(li.iterator());
1706          while (i.hasNext()) {
1707              System.out.println(i.next());
1708          }
1709      }
1710  }
```

```
1711
1712
1713 class foo {
1714     public static void aaarg() { System.out.println("aaarg"); }
1715 }
1716
1717 public class beta {
1718
1719     static Random rand = new Random();
1720
1721     private static final double gam(int x) {
1722         double result = 0;
1723         for (int i = 1; i <= x; i++) {
1724             result += Math.log(rand.nextDouble());
1725         }
1726         return result;
1727     }
1728
1729     public static double draw(int a, int b) {
1730         return gam(a) / (gam(a) + gam(b));
1731     }
1732
1733     public static void main(String[] args) {
1734         for (int i = 0; i < 100000; i++) {
1735             System.out.println(draw(2, 5));
1736         }
1737     }
1738 }
1739
1740 class Thing {
1741     public Thing() {
1742         Runtime.getRuntime().addShutdownHook(new Thread() {
1743                 public void run() { flush(); }
1744             });
1745     }
1746
1747     public void flush() { /* do some deferred action */ }
1748
1749     public static void main(String[] args) {
1750         Thing t;
1751         while (true) { t = new Thing(); }
1752     }
1753 }
1754
1755 class crash {
1756     public static void main(String[] args) {
1757         Object[] o = new Object[]{};
1758         while (null != o) { o = new Object[]{o}; }
1759     }
1760 }
1761
1762 class GenericDemo {
1763     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
    ) {
1764         return new Iterator<T>() {
1765             private Iterator<? extends T> buffer = null;
1766
1767             public void remove() {
1768                 throw new UnsupportedOperationException("Not supported");
1769             }
1770
1771             public T next() {
1772                 if (!hasNext()) {
1773                     throw new NoSuchElementException("No more elements");
1774                 }
1775                 return buffer.next();
1776             }
1777
1778             public boolean hasNext() {
1779                 while (source.hasNext()
1780                         && (null == buffer
1781                             || !buffer.hasNext())) {
```



```
1782                     buffer = source.next();
1783                 }
1784
1785                 return buffer.hasNext();
1786             }
1787         };
1788     }
1789
1790     public static void main(String[] args) {
1791         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
1792         List<String> l = new ArrayList<String>();
1793         l.add("foo"); l.add("bar");
1794         li.add(l.iterator());
1795
1796         l = new ArrayList<String>();
1797         l.add("baz"); l.add("quux");
1798         li.add(l.iterator());
1799
1800         Iterator<String> i = collapse(li.iterator());
1801         while (i.hasNext()) {
1802             System.out.println(i.next());
1803         }
1804     }
1805 }
1806
1807
1808 class foo {
1809     public static void aaarg() { System.out.println("aaarg"); }
1810 }
1811
1812 public class beta {
1813
1814     static Random rand = new Random();
1815
1816     private static final double gam(int x) {
1817         double result = 0;
1818         for (int i = 1; i <= x; i++) {
1819             result += Math.log(rand.nextDouble());
1820         }
1821         return result;
1822     }
1823
1824     public static double draw(int a, int b) {
1825         return gam(a) / (gam(a) + gam(b));
1826     }
1827
1828     public static void main(String[] args) {
1829         for (int i = 0; i < 100000; i++) {
1830             System.out.println(draw(2, 5));
1831         }
1832     }
1833 }
1834
1835 class Thing {
1836     public Thing() {
1837         Runtime.getRuntime().addShutdownHook(new Thread() {
1838                 public void run() { flush(); }
1839             });
1840     }
1841
1842     public void flush() { /* do some deferred action */ }
1843
1844     public static void main(String[] args) {
1845         Thing t;
1846         while (true) { t = new Thing(); }
1847     }
1848 }
1849
1850 class crash {
1851     public static void main(String[] args) {
1852         Object[] o = new Object[]{};
1853         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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{};
```

```
2424         while (null != o) { o = new Object[]{o}; }
2425     }
2426 }
2427
2428 class GenericDemo {
2429     public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
     ) {
2430         return new Iterator<T>() {
2431             private Iterator<? extends T> buffer = null;
2432
2433             public void remove() {
2434                 throw new UnsupportedOperationException("Not supported");
2435             }
2436
2437             public T next() {
2438                 if (!hasNext()) {
2439                     throw new NoSuchElementException("No more elements");
2440                 }
2441                 return buffer.next();
2442             }
2443
2444             public boolean hasNext() {
2445                 while (source.hasNext()
2446                        && (null == buffer
2447                            || !buffer.hasNext())) {
2448                     buffer = source.next();
2449                 }
2450
2451                 return buffer.hasNext();
2452             }
2453         };
2454     }
2455
2456     public static void main(String[] args) {
2457         List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2458         List<String> l = new ArrayList<String>();
2459         l.add("foo"); l.add("bar");
2460         li.add(l.iterator());
2461
2462         l = new ArrayList<String>();
2463         l.add("baz"); l.add("quux");
2464         li.add(l.iterator());
2465
2466         Iterator<String> i = collapse(li.iterator());
2467         while (i.hasNext()) {
2468             System.out.println(i.next());
2469         }
2470     }
2471 }
2472
2473
2474 class foo {
2475     public static void aaarg() { System.out.println("aaarg"); }
2476 }
2477
2478 public class beta {
2479
2480     static Random rand = new Random();
2481
2482     private static final double gam(int x) {
2483         double result = 0;
2484         for (int i = 1; i <= x; i++) {
2485             result += Math.log(rand.nextDouble());
2486         }
2487         return result;
2488     }
2489
2490     public static double draw(int a, int b) {
2491         return gam(a) / (gam(a) + gam(b));
2492     }
2493
2494     public static void main(String[] args) {
```



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

```
}


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[]{};
        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 (!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>>();
        List<String> l = new ArrayList<String>();
        l.add("foo"); l.add("bar");
        li.add(l.iterator());

        l = new ArrayList<String>();
        l.add("baz"); l.add("quux");
        li.add(l.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 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[]{};
```

```
2709          while (null != o) { o = new Object[]{o}; }
2710      }
2711  }
2712
2713  class GenericDemo {
2714      public static <T> Iterator<T> collapse(final Iterator<? extends Iterator<? extends T>> source
      ) {
2715          return new Iterator<T>() {
2716              private Iterator<? extends T> buffer = null;
2717
2718              public void remove() {
2719                  throw new UnsupportedOperationException("Not supported");
2720              }
2721
2722              public T next() {
2723                  if (!hasNext()) {
2724                      throw new NoSuchElementException("No more elements");
2725                  }
2726                  return buffer.next();
2727              }
2728
2729              public boolean hasNext() {
2730                  while (source.hasNext()
2731                          && (null == buffer
2732                              || !buffer.hasNext())) {
2733                      buffer = source.next();
2734                  }
2735
2736                  return buffer.hasNext();
2737              }
2738          };
2739      }
2740
2741      public static void main(String[] args) {
2742          List<Iterator<String>> li = new ArrayList<Iterator<String>>();
2743          List<String> l = new ArrayList<String>();
2744          l.add("foo"); l.add("bar");
2745          li.add(l.iterator());
2746
2747          l = new ArrayList<String>();
2748          l.add("baz"); l.add("quux");
2749          li.add(l.iterator());
2750
2751          Iterator<String> i = collapse(li.iterator());
2752          while (i.hasNext()) {
2753              System.out.println(i.next());
2754          }
2755      }
2756  }
2757
end
```