```java
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()
```

```java
                       && (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) {
```

```java
        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[]{};
```

```java
        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) {
```

```java
        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());
        }
    }
```

```java
}


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
```

```java
                           || !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++) {
```

```java
            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());
        }
    }
}
```

```java

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())) {
```

```java
                    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++) {
```

```java
            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());
        }
    }
}
```

```java

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())) {
```

```java
                    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());
        }
    }
}
```

```java

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())) {
```

```java
                    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++) {
```

```java
            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());
        }
    }
}
```

```java

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())) {
```

```java
                    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) {
```

```java
        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());
        }
    }
```

```java
}

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
```

```java
                           || !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) {
```

```java
        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());
        }
    }
```

```java
}

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
```

```java
                           || !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());
        }
    }
```

```java
}


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
```

```java
                           || !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());
        }
    }
}
```

end