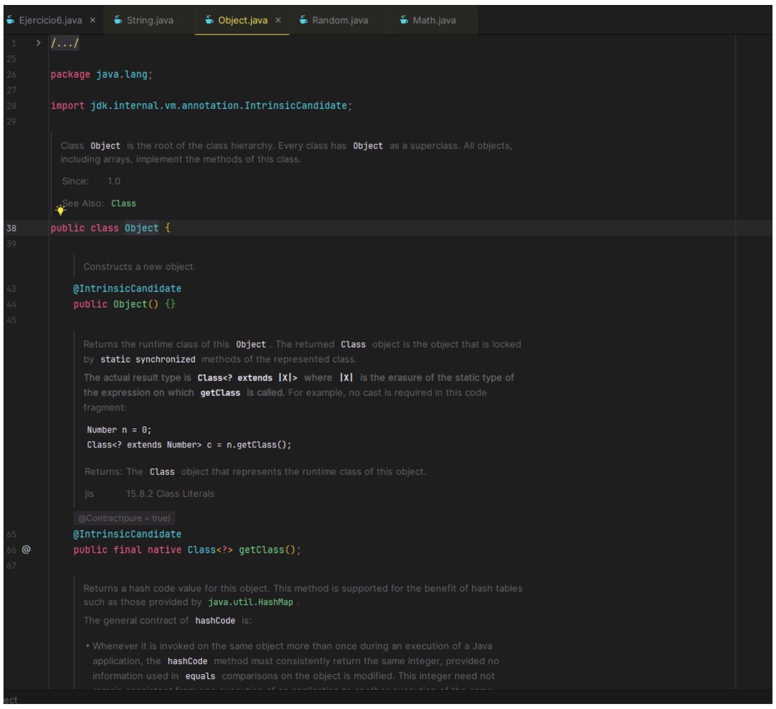
Object:



```java
/.../

package java.lang;

import jdk.internal.vm.annotation.IntrinsicCandidate;
```

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Since: 1.0

See Also: `Class`

```java
public class Object {
```

Constructs a new object.

```java
    @IntrinsicCandidate
    public Object() {}
```

Returns the runtime class of this `Object` . The returned `Class` object is the object that is locked by `static synchronized` methods of the represented class.

The actual result type is `Class<? extends |X|>` where `|X|` is the erasure of the static type of the expression on which `getClass` is called. For example, no cast is required in this code fragment:

```java
Number n = 0;
Class<? extends Number> c = n.getClass();
```

Returns: The `Class` object that represents the runtime class of this object.

jls 15.8.2 Class Literals

```java
    @Contract(pure = true)
    @IntrinsicCandidate
    public final native Class<?> getClass();
```

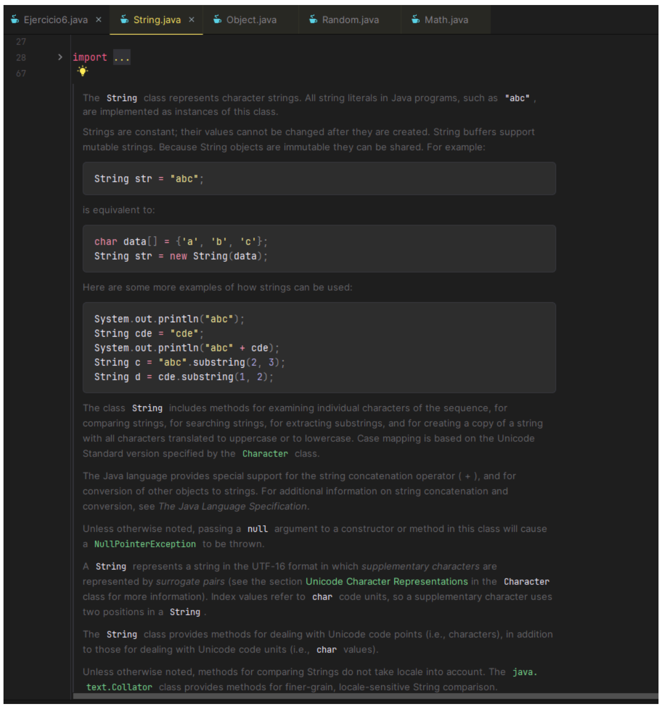
Returns a hash code value for this object. This method is supported for the benefit of hash tables such as those provided by `java.util.HashMap` .

The general contract of `hashCode` is:

- Whenever it is invoked on the same object more than once during an execution of a Java application, the `hashCode` method must consistently return the same integer, provided no information used in `equals` comparisons on the object is modified. This integer need not

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Object.html

String:



```java
import ...
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"` , are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```java
String str = "abc";
```

is equivalent to:

```java
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```java
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2, 3);
String d = cde.substring(1, 2);
```

The class `String` includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the `Character` class.

The Java language provides special support for the string concatenation operator (+), and for conversion of other objects to strings. For additional information on string concatenation and conversion, see *The Java Language Specification*.

Unless otherwise noted, passing a `null` argument to a constructor or method in this class will cause a `NullPointerException` to be thrown.

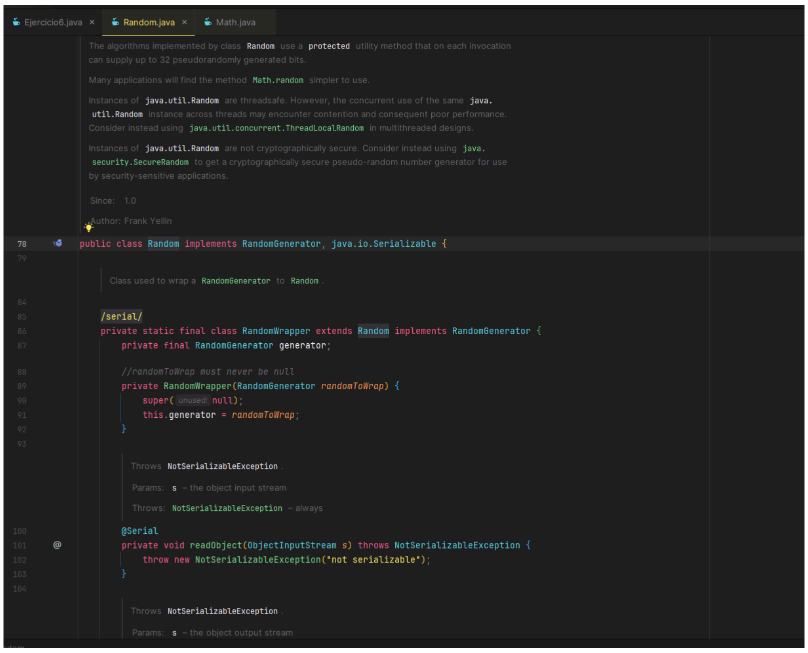
A `String` represents a string in the UTF-16 format in which *supplementary characters* are represented by *surrogate pairs* (see the section Unicode Character Representations in the `Character` class for more information). Index values refer to `char` code units, so a supplementary character uses two positions in a `String` .

The `String` class provides methods for dealing with Unicode code points (i.e., characters), in addition to those for dealing with Unicode code units (i.e., `char` values).

Unless otherwise noted, methods for comparing Strings do not take locale into account. The `java.text.Collator` class provides methods for finer-grain, locale-sensitive String comparison.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html

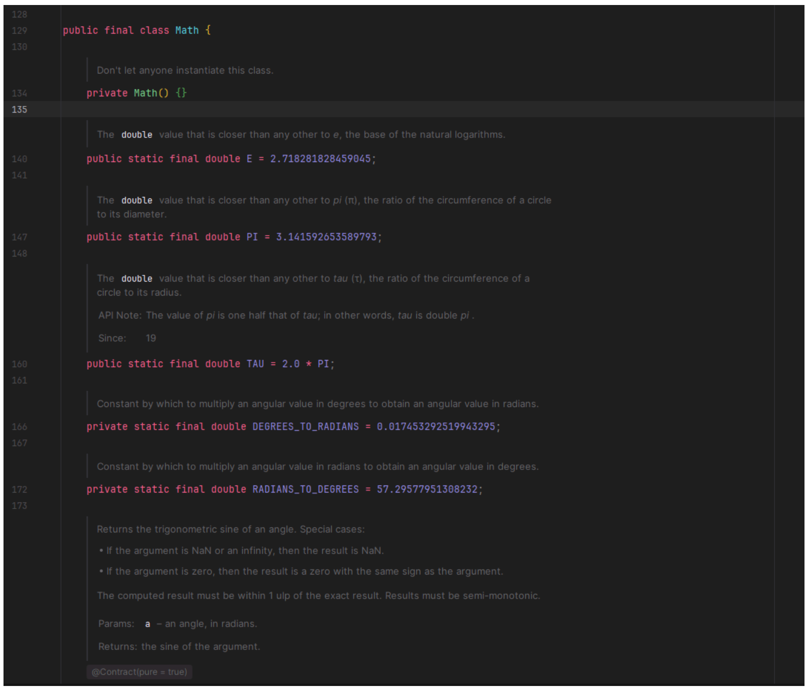
Random:



```
The algorithms implemented by class Random use a protected utility method that on each invocation
can supply up to 32 pseudorandomly generated bits.

Many applications will find the method Math.random simpler to use.

Instances of java.util.Random are threadsafe. However, the concurrent use of the same java.
util.Random instance across threads may encounter contention and consequent poor performance.
Consider instead using java.util.concurrent.ThreadLocalRandom in multithreaded designs.

Instances of java.util.Random are not cryptographically secure. Consider instead using java.
security.SecureRandom to get a cryptographically secure pseudo-random number generator for use
by security-sensitive applications.

Since: 1.0

Author: Frank Yellin
public class Random implements RandomGenerator, java.io.Serializable {

    Class used to wrap a RandomGenerator to Random .

    /serial/
    private static final class RandomWrapper extends Random implements RandomGenerator {
        private final RandomGenerator generator;

        //randomToWrap must never be null
        private RandomWrapper(RandomGenerator randomToWrap) {
            super( unused: null);
            this.generator = randomToWrap;
        }

        Throws NotSerializableException .

        Params: s – the object input stream

        Throws: NotSerializableException – always

        @Serial
        private void readObject(ObjectInputStream s) throws NotSerializableException {
            throw new NotSerializableException("not serializable");
        }

        Throws NotSerializableException .

        Params: s – the object output stream
```

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Random.html

Math:



```
public final class Math {

    Don't let anyone instantiate this class.
    private Math() {}

    The double value that is closer than any other to e, the base of the natural logarithms.
    public static final double E = 2.718281828459045;

    The double value that is closer than any other to pi (π), the ratio of the circumference of a circle
    to its diameter.
    public static final double PI = 3.141592653589793;

    The double value that is closer than any other to tau (τ), the ratio of the circumference of a
    circle to its radius.

    API Note: The value of pi is one half that of tau; in other words, tau is double pi .

    Since: 19
    public static final double TAU = 2.0 * PI;

    Constant by which to multiply an angular value in degrees to obtain an angular value in radians.
    private static final double DEGREES_TO_RADIANS = 0.017453292519943295;

    Constant by which to multiply an angular value in radians to obtain an angular value in degrees.
    private static final double RADIANS_TO_DEGREES = 57.29577951308232;

    Returns the trigonometric sine of an angle. Special cases:
    • If the argument is NaN or an infinity, then the result is NaN.
    • If the argument is zero, then the result is a zero with the same sign as the argument.
    The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

    Params: a – an angle, in radians.

    Returns: the sine of the argument.
    @Contract(pure = true)
```

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html