Package java.lang

Provides classes that are fundamental to the design of the Java programming language.

See: Description

提供Java语言设计的基础类

Interface Summary

Interface	Description
Appendable	An object to which char sequences and values can be appended.
AutoCloseable	An object that may hold resources (such as file or socket handles) until it is closed.
CharSequence	A CharSequence is a readable sequence of char values.
Cloneable	A class implements the Cloneable interface to indicate to the **Object.clone()** method that it is legal for that method to make a field-for-field copy of instances of that class.
Comparable<T>	This interface imposes a total ordering on the objects of each class that implements it.
Iterable<T>	Implementing this interface allows an object to be the target of the "for-each loop" statement.
Readable	A Readable is a source of characters.
Runnable	The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
Thread.UncaughtExceptionHandler	Interface for handlers invoked when a Thread abruptly terminates due to an uncaught exception.

Class Summary

Class	Description
Boolean	The Boolean class wraps a value of the primitive type boolean in an object.
Byte	The Byte class wraps a value of primitive type byte in an object.
Character	The Character class wraps a value of the primitive type char in an object.
Character.Subset	Instances of this class represent particular subsets of the Unicode character set.
Character.UnicodeBlock	

	A family of character subsets representing the character blocks in the Unicode specification.
Class<T> 表示运行应用中类型和接口的实例	Instances of the class `Class` represent classes and interfaces in a running Java application.
ClassLoader	A class loader is an object that is responsible for loading classes.
ClassValue<T>	Lazily associate a computed value with (potentially) every type.
Compiler	The `Compiler` class is provided to support Java-to-native-code compilers and related services.
Double	The `Double` class wraps a value of the primitive type `double` in an object.
Enum<E extends **Enum**<E>>	This is the common base class of all Java language enumeration types.
Float	The `Float` class wraps a value of primitive type `float` in an object.
InheritableThreadLocal<T>	This class extends `ThreadLocal` to provide inheritance of values from parent thread to child thread: when a child thread is created, the child receives initial values for all inheritable thread-local variables for which the parent has values.
Integer	The `Integer` class wraps a value of the primitive type `int` in an object.
Long	The `Long` class wraps a value of the primitive type `long` in an object.
Math	The class `Math` contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
Number	The abstract class `Number` is the superclass of platform classes representing numeric values that are convertible to the primitive types `byte`, `double`, `float`, `int`, `long`, and `short`.
Object 是类层次结构树的根	Class `Object` is the root of the class hierarchy.
Package	`Package` objects contain version information about the implementation and specification of a Java package.
Process	The **`ProcessBuilder.start()`** and **`Runtime.exec`** methods create a native process and return an instance of a subclass of `Process` that can be used to control the process and obtain information about it.
ProcessBuilder	This class is used to create operating system processes.

ProcessBuilder.Redirect	Represents a source of subprocess input or a destination of subprocess output.
Runtime	Every Java application has a single instance of class `Runtime` that allows the application to interface with the environment in which the application is running.
RuntimePermission	This class is for runtime permissions.
SecurityManager	The security manager is a class that allows applications to implement a security policy.
Short	The `Short` class wraps a value of primitive type `short` in an object.
StackTraceElement	An element in a stack trace, as returned by **`Throwable.getStackTrace()`**.
StrictMath	The class `StrictMath` contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
String	The `String` class represents character strings.
StringBuffer	A thread-safe, mutable sequence of characters.
StringBuilder	A mutable sequence of characters.
System	The `System` class contains several useful class fields and methods.
Thread	A *thread* is a thread of execution in a program.
ThreadGroup	A thread group represents a set of threads.
ThreadLocal<T>	This class provides thread-local variables.
Throwable	The `Throwable` class is the superclass of all errors and exceptions in the Java language.
Void	The `Void` class is an uninstantiable placeholder class to hold a reference to the `Class` object representing the Java keyword void.

Enum Summary

Enum	Description
Character.UnicodeScript	A family of character subsets representing the character scripts defined in the ***Unicode Standard Annex #24: Script Names***.
ProcessBuilder.Redirect.Type	The type of a **`ProcessBuilder.Redirect`**.
Thread.State	A thread state.

Exception Summary

Exception	Description
ArithmeticException	Thrown when an exceptional arithmetic condition has occurred.
ArrayIndexOutOfBoundsException	Thrown to indicate that an array has been accessed with an illegal index.
ArrayStoreException	Thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects.
ClassCastException	Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance.
ClassNotFoundException	Thrown when an application tries to load in a class through its string name using: The `forName` method in class `Class`.
CloneNotSupportedException	Thrown to indicate that the `clone` method in class `Object` has been called to clone an object, but that the object's class does not implement the `Cloneable` interface.
EnumConstantNotPresentException	Thrown when an application tries to access an enum constant by name and the enum type contains no constant with the specified name.
Exception	The class `Exception` and its subclasses are a form of `Throwable` that indicates conditions that a reasonable application might want to catch.
IllegalAccessException	An IllegalAccessException is thrown when an application tries to reflectively create an instance (other than an array), set or get a field, or invoke a method, but the currently executing method does not have access to the definition of the specified class, field, method or constructor.
IllegalArgumentException	Thrown to indicate that a method has been passed an illegal or inappropriate argument.
IllegalMonitorStateException	Thrown to indicate that a thread has attempted to wait on an object's monitor or to notify other threads waiting on an object's monitor without owning the specified monitor.
IllegalStateException	Signals that a method has been invoked at an illegal or inappropriate time.
IllegalThreadStateException	Thrown to indicate that a thread is not in an appropriate state for the requested operation.
IndexOutOfBoundsException	Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of

	as to an array, to a string, or to a vector) is out of range.
InstantiationException	Thrown when an application tries to create an instance of a class using the `newInstance` method in class `Class`, but the specified class object cannot be instantiated.
InterruptedException	Thrown when a thread is waiting, sleeping, or otherwise occupied, and the thread is interrupted, either before or during the activity.
NegativeArraySizeException	Thrown if an application tries to create an array with negative size.
NoSuchFieldException	Signals that the class doesn't have a field of a specified name.
NoSuchMethodException	Thrown when a particular method cannot be found.
NullPointerException	Thrown when an application attempts to use `null` in a case where an object is required.
NumberFormatException	Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.
ReflectiveOperationException	Common superclass of exceptions thrown by reflective operations in core reflection.
RuntimeException	`RuntimeException` is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.
SecurityException	Thrown by the security manager to indicate a security violation.
StringIndexOutOfBoundsException	Thrown by `String` methods to indicate that an index is either negative or greater than the size of the string.
TypeNotPresentException	Thrown when an application tries to access a type using a string representing the type's name, but no definition for the type with the specified name can be found.
UnsupportedOperationException	Thrown to indicate that the requested operation is not supported.

Error Summary

Error	Description
AbstractMethodError	Thrown when an application tries to call an abstract method.
AssertionError	Thrown to indicate that an assertion has failed.

BootstrapMethodError	Thrown to indicate that an `invokedynamic` instruction has failed to find its bootstrap method, or the bootstrap method has failed to provide a **call site** with a **target** of the correct **method type**.
ClassCircularityError	Thrown when the Java Virtual Machine detects a circularity in the superclass hierarchy of a class being loaded.
ClassFormatError	Thrown when the Java Virtual Machine attempts to read a class file and determines that the file is malformed or otherwise cannot be interpreted as a class file.
Error	An `Error` is a subclass of `Throwable` that indicates serious problems that a reasonable application should not try to catch.
ExceptionInInitializerError	Signals that an unexpected exception has occurred in a static initializer.
IllegalAccessError	Thrown if an application attempts to access or modify a field, or to call a method that it does not have access to.
IncompatibleClassChangeError	Thrown when an incompatible class change has occurred to some class definition.
InstantiationError	Thrown when an application tries to use the Java `new` construct to instantiate an abstract class or an interface.
InternalError	Thrown to indicate some unexpected internal error has occurred in the Java Virtual Machine.
LinkageError	Subclasses of `LinkageError` indicate that a class has some dependency on another class; however, the latter class has incompatibly changed after the compilation of the former class.
NoClassDefFoundError	Thrown if the Java Virtual Machine or a `ClassLoader` instance tries to load in the definition of a class (as part of a normal method call or as part of creating a new instance using the `new` expression) and no definition of the class could be found.
NoSuchFieldError	Thrown if an application tries to access or modify a specified field of an object, and that object no longer has that field.
NoSuchMethodError	Thrown if an application tries to call a specified method of a class (either static or instance), and that class no longer has a definition of that method.
OutOfMemoryError	Thrown when the Java Virtual Machine cannot allocate an object because it is out of memory, and no more memory could be made available by the garbage collector.
StackOverflowError	

	Thrown when a stack overflow occurs because an application recurses too deeply.
ThreadDeath	An instance of `ThreadDeath` is thrown in the victim thread when the (deprecated) **`Thread.stop()`** method is invoked.
UnknownError	Thrown when an unknown but serious exception has occurred in the Java Virtual Machine.
UnsatisfiedLinkError	Thrown if the Java Virtual Machine cannot find an appropriate native-language definition of a method declared `native`.
UnsupportedClassVersionError	Thrown when the Java Virtual Machine attempts to read a class file and determines that the major and minor version numbers in the file are not supported.
VerifyError	Thrown when the "verifier" detects that a class file, though well formed, contains some sort of internal inconsistency or security problem.
VirtualMachineError	Thrown to indicate that the Java Virtual Machine is broken or has run out of resources necessary for it to continue operating.

Annotation Types Summary

Annotation Type	Description
Deprecated	A program element annotated @Deprecated is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists.
FunctionalInterface	An informative annotation type used to indicate that an interface type declaration is intended to be a *functional interface* as defined by the Java Language Specification.
Override	Indicates that a method declaration is intended to override a method declaration in a supertype.
SafeVarargs	A programmer assertion that the body of the annotated method or constructor does not perform potentially unsafe operations on its varargs parameter.
SuppressWarnings	Indicates that the named compiler warnings should be suppressed in the annotated element (and in all program elements contained in the annotated element).

Package java.lang Description

Provides classes that are fundamental to the design of the Java programming language. The most important classes are `Object`, which is the root of the class hierarchy, and `Class`, instances of which represent classes at run time.

Frequently it is necessary to represent a value of primitive type as if it were an object. The wrapper classes `Boolean`, `Character`, `Integer`, `Long`, `Float`, and `Double` serve this purpose. An object of type `Double`, for example, contains a field whose type is double, representing

that value in such a way that a reference to it can be stored in a variable of reference type. These classes also provide a number of methods for converting among primitive values, as well as supporting such standard methods as equals and hashCode. The `Void` class is a non-instantiable class that holds a reference to a `Class` object representing the type void.

The class `Math` provides commonly used mathematical functions such as sine, cosine, and square root. The classes `String`, `StringBuffer`, and `StringBuilder` similarly provide commonly used operations on character strings.

Classes `ClassLoader`, `Process`, `ProcessBuilder`, `Runtime`, `SecurityManager`, and `System` provide "system operations" that manage the dynamic loading of classes, creation of external processes, host environment inquiries such as the time of day, and enforcement of security policies.

Class `Throwable` encompasses objects that may be thrown by the `throw` statement. Subclasses of `Throwable` represent errors and exceptions.

Character Encodings

The specification of the `java.nio.charset.Charset` class describes the naming conventions for character encodings as well as the set of standard encodings that must be supported by every implementation of the Java platform.

Since:

JDK1.0