

# Java.util Package in Java

## Java.util Package

It contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

**Following are the Important Classes in Java.util package :**

1. **AbstractCollection**: This class provides a skeletal implementation of the Collection interface, to minimize the effort required to implement this interface.
2. **AbstractList**: This class provides a skeletal implementation of the List interface to minimize the effort required to implement this interface backed by a “random access” data store (such as an array).
3. **AbstractMap<K,V>**: This class provides a skeletal implementation of the Map interface, to minimize the effort required to implement this interface.
4. **AbstractMap.SimpleEntry<K,V>**: An Entry maintaining a key and a value.
5. **AbstractMap.SimpleImmutableEntry<K,V>**: An Entry maintaining an immutable key and value.
6. **AbstractQueue**: This class provides skeletal implementations of some Queue operations.
7. **AbstractSequentialList**: This class provides a skeletal implementation of the List interface to minimize the effort required to implement this interface backed by a “sequential access” data store (such as a linked list).
8. **AbstractSet**: This class provides a skeletal implementation of the Set interface to minimize the effort required to implement this interface.
9. **ArrayDeque**: Resizable-array implementation of the Deque interface.
10. **ArrayList**: Resizable-array implementation of the List interface.
11. **Arrays**: This class contains various methods for manipulating arrays (such as sorting and searching).
12. **BitSet**: This class implements a vector of bits that grows as needed.
13. **Calendar**: The Calendar class is an abstract class that provides methods for converting between a specific instant in time and a set of calendar fields such as YEAR, MONTH, DAY\_OF\_MONTH, HOUR, and so on, and for manipulating the calendar fields, such as getting the date of the next week.
14. **Collections**: This class consists exclusively of static methods that operate on or return collections.
15. **Currency**: Represents a currency.
16. **Date**: The class Date represents a specific instant in time, with millisecond precision.
17. **Dictionary<K,V>**: The Dictionary class is the abstract parent of any class, such as Hashtable, which maps keys to values.
18. **EnumMap,V>**: A specialized Map implementation for use with enum type keys.
19. **EnumSet**: A specialized Set implementation for use with enum types.
20. **EventListenerProxy**: An abstract wrapper class for an EventListener class which associates a set of additional parameters with the listener.

21. **EventObject**: The root class from which all event state objects shall be derived.
22. **FormattableFlags**: FormattableFlags are passed to the Formattable.formatTo() method and modify the output format for Formattables.
23. **Formatter**: An interpreter for printf-style format strings.
24. **GregorianCalendar**: GregorianCalendar is a concrete subclass of Calendar and provides the standard calendar system used by most of the world.
25. **HashMap<K,V>**: Hash table based implementation of the Map interface.
26. **HashSet**: This class implements the Set interface, backed by a hash table (actually a HashMap instance).
27. **Hashtable<K,V>**: This class implements a hash table, which maps keys to values.
28. **IdentityHashMap<K,V>**: This class implements the Map interface with a hash table, using reference-equality in place of object-equality when comparing keys (and values).
29. **LinkedHashMap<K,V>**: Hash table and linked list implementation of the Map interface, with predictable iteration order.
30. **LinkedHashSet**: Hash table and linked list implementation of the Set interface, with predictable iteration order.
31. **LinkedList**: Doubly-linked list implementation of the List and Deque interfaces.
32. **ListResourceBundle**: ListResourceBundle is an abstract subclass of ResourceBundle that manages resources for a locale in a convenient and easy to use list.
33. **Locale** – [Set 1](#), [Set 2](#): A Locale object represents a specific geographical, political, or cultural region.
34. **Locale.Builder**: Builder is used to build instances of Locale from values configured by the setters.
35. **Objects**: This class consists of static utility methods for operating on objects.
36. **Observable**: This class represents an observable object, or “data” in the model-view paradigm.
37. **PriorityQueue**: An unbounded priority queue based on a priority heap.
38. **Properties**: The Properties class represents a persistent set of properties.
39. **PropertyPermission**: This class is for property permissions.
40. **PropertyResourceBundle**: PropertyResourceBundle is a concrete subclass of ResourceBundle that manages resources for a locale using a set of static strings from a property file.
41. **Random**: An instance of this class is used to generate a stream of pseudorandom numbers.
42. **ResourceBundle**: Resource bundles contain locale-specific objects.
43. **ResourceBundle.Control**: ResourceBundle.Control defines a set of callback methods that are invoked by the ResourceBundle.getBundle factory methods during the bundle loading process.
44. **Scanner**: A simple text scanner which can parse primitive types and strings using regular expressions.
45. **ServiceLoader**: A simple service-provider loading facility.
46. **SimpleTimeZone**: SimpleTimeZone is a concrete subclass of TimeZone that represents a time zone for use with a Gregorian calendar.
47. **Stack**: The Stack class represents a last-in-first-out (LIFO) stack of objects.
48. **StringTokenizer**: The string tokenizer class allows an application to break a string into tokens.

- 49. **Timer**: A facility for threads to schedule tasks for future execution in a background thread.
- 50. **TimerTask**: A task that can be scheduled for one-time or repeated execution by a Timer.
- 51. **TimeZone**: TimeZone represents a time zone offset, and also figures out daylight savings.
- 52. **TreeMap<K,V>**: A Red-Black tree based NavigableMap implementation.
- 53. **TreeSet**: A NavigableSet implementation based on a TreeMap.
- 54. **UUID**: A class that represents an immutable universally unique identifier (UUID).
- 55. **Vector**: The Vector class implements a growable array of objects.
- 56. **WeakHashMap<K,V>**: Hash table based implementation of the Map interface, with weak keys.