COLLECTIONS IN JAVA

java.util.\*;

Collection framework used to store collection of objects provided with two things, Collection Interface and Collection Class.

Collection Interface

Core interface used to store a collection of objects.

ArrayList

ArrayList Class:

class Array extends AbstractList class implements Serializable, Clonable

it is a dynamic array where we increase or decrease its size at runtime.

faster retrieval and slower in insertion and deletion.

the default capacity of ArrayList is 10

retainAll() – removes the element from the target collection that is not contained on the specific collection

LinkedList

LinkedList extends AbstractSequentialList implements List, Dequeue, Serializable, Clonable

LinkedList similar to ArrayList.

faster in insertion and deletion and slower in retrieval

Vector

Vector extends AbstractList implements List, Serializable, Clonable

Vector is similar to ArrayList, but it is synchronized

it is a legacy class

the default capacity is 10

HashSet

HashSet extends AbstractSet implements Set, Serializable, Clonable

HashSet prints the elements in random order

default capacity is 16

LinkedHashSet

LinkedHashSet extends HashSet implements Set, Serializable, Clonable

it will print in the order in which we have inserted

TreeSet

TreeSet extends HashSet implements SortedSet, Clonable, Serializable

it is used to print the elements in the sorted order

no combination of String and numbers

Collections Class

It contains static methods and algorithms which supports the util package.

Methods:

1. static int bainarySearch(List l, int val)
2. static void sort(List l)
3. static void sort(List l, Comparator c)
4. static void shuffle(List l)
5. static void swap(List l, int old, int new)
6. static object min(Collections c)
7. static object max(Collections c)
8. static void copy(List destination, List source)
9. static void fill(List l, int val)
10. static int frequency(Collection c, Object o)
11. static boolean replaceAll(List l, Object old, Object new)
12. static void reverse(List l)
13. static set Singleton(Object o)
14. static list singletonList(Object o)
15. static list synchronisedList(List l)
16. static list unmodifiableList(List l)
17. static Boolean disjoint(Collections e1, Collections e2)

Comparable & Comparator:

1. Comparable provides single sorting sequence whereas Comparator provides multiple sorting sequence.
2. Comparable will affect the original class whereas Comparator won’t affect the original class
3. Comparable contains one method public int compareTo(Object obj) and Comparator contains public int compare(Object o1, Object o2)
4. Comparable present in java.lang package whereas Comparator is present in java.util package
5. We do sorting using Collections.sort(List l) for Comaparable and Collections.sort(List l, Comparator c) for Comaparator

Map Interface:

It contains collection of objects as a unique key value pair.

Map is unordered and doesn’t allow duplicate elements.

Map Hierarchy 🡪(i)AbstractMap🡪(e)HashMap🡪(e)LinkedHashMap

🡪(e)Sorted Map🡪(i)TreeMap

HashMap Class:

HashMap extends AbstractMap implements Map, Serializable, Clonable

Contains collections of key-value pairs and prints in random order

Default capacity is 16

LinkedHashMap:

LinkedHashMap extends HashMap implements Map, Clonable, Serializable

Contains key-value pairs in the order which we have inserted

TreeMap Class:

TreeMap extends AbstractMap implements SortedMap, Clonable, Serializable

Contains key-value pairs and prints in sorted order based on the key

The best is HashMap.

Methods:

1. Set entrySet(): converts both key and values to the set interface
2. Set keyset(): converts only the key to the set interface

Map.Entry interface:

It is used to get the key and value separately

1. Object getKey(): to get the key
2. Object getValue(): to get the value

Hashtable:

It is a legacy class, similar to HashMap but it is synchronized.

1. Enumeration elements(): converts both key and value
2. Enumeration keys(): converts only keys

Properties Class:

It is a sub class of Hashtable, it is synchronized.

Contains collection of key-value pairs, print in random order but both key and value should be in the string.

Date Class:

It is used to print current date and time

Constructor-

1. Date(){}: current date and time
2. Date(long millisecond){}: calculate date and time from Jan 1st 1970

Methods-

1. boolean after(Date d)
2. boolean before(Date d)
3. int compareTo(Date d)
4. long getTime()
5. void setTime(long msec)

Calendar Class:

It is an Abstract Class used to extract useful information from date and time components like hour, minute, second, month, year etc.

Constructor:

1. Calendar()

Method:

1. static Calendar getInstance(): used to create object for calendar class
2. public void add(int field, int value)
3. public void set(int field, int value)
4. public boolean after(Object o)
5. public boolean before(Object o)
6. public int get(int field)
7. public int getActualMaximum(int feild)
8. public int getActualMinimum(int feild)
9. public int getFirstDayOfWeek()
10. public final Date getTime(): converts calendar to date
11. public void setTime(Date d): converts Date to Calendar
12. public int getWeeksInWeekYear()
13. public void setWeekDate(int year, int week\_of\_year, int day\_of\_year)
14. public abstract int getMinimum(int field)
15. public abstract int getMaximum(int field)

Constants:

1. public static final int HOUR;
2. public static final int MONTH;
3. public static final int YEAR;
4. public static final int MINUTE;
5. public static final int SECOND;
6. public static final int ERA;
7. public static final int HOUR\_OF\_DAY;
8. public static final int WEEK\_OF\_MONTH;
9. public static final int WEEK\_OF\_YEAR;
10. public static final int MILLISECOND;
11. public static final int DATE;
12. public static final int DAY\_OF\_WEEK;
13. public static final int DAY\_OF\_MONTH;
14. public static final int MONDAY; to public static final int SUNDAY;
15. public static final int JANUARY; to public static final int DECEMBER;

GregorianCalendar:

It is concrete implementation of Calendar class.

Constructor:

1. GregorianCalendar(): returns current date and time
2. GregorianCalendar(int year, int month, int date)
3. GregorianCalendar(int year, int month, int date, int hour, int minute)
4. GregorianCalendar(int year, int month, int date, int hour, int minute, int second)

Methods:

1. boolean isLeapYear(int year)

DateFormat Class:

It is an abstract class used to format and parse the Date in different style.

It is present in java.text.\* package.

Formatting is converting from Date to String

Parsing is Converting from String to Date

Constructor:

1. DateFormat()

Constants:

1. public static final int short;
2. public static final int medium;
3. public static final int long;
4. public static final int full;

Methods:

1. static String format(Date d)
2. Date parse(String s)
3. static DateFormat getInstance(): prints date and time in short format
4. static DateFormat getDateInstance(): print only date in medium format
5. static DateFormat getDateInstance(int date\_style): prints date in desired style
6. static DateFormat getTimeInstance(): prints only time in medium style
7. static DateFormat getTimeInstance(int time\_style): prints time in desired style
8. static DateFormat getDateTimeInstance(): prints both date and time in medium style
9. static DateFormat getDateTimeInstance(int date\_style, int time\_style): prints both in desired style

SimpleDateFormat Class:

It is a concrete implementation of DateFormat class where we can define our own styles.

It is present in java.text.\* package.

Constructor:

1. SimpleDateFormat()
2. SimpleDateFormat(String pattern)

Patterns:

1. G: represents era in text
2. y: represents year in number
3. M: represents month in text
4. w: represents week in year in number
5. W: represents week in month in number
6. D: represents day in year in number
7. d: represents day in month in number
8. H: represents hours in a day
9. m: represents minutes
10. s: represents seconds
11. S: represents milliseconds
12. z: represents time zone