JA111

Day-13

Date and Time API in java

The data and time related classes, interfaces and enums are part of java.time package. Some important classes of this package are

1. **The LocalDate class**
2. **The LocalTime class**
3. **The LocalDateTime class**
4. **The DateTimeFormatter class of package java.time.format package**
5. **The Duration class**
6. **The Period class**
7. **The ChronoUnit Enum**

The LocalDate class

❑ **The LocalDate represents a date in ISO format (yyyy-MM-dd) without time. It can be used to store dates like birthdays and paydays.**

❑ **This class is immutable and thread-safe.**

❑ **This class is final so not possible to create subclass of the same**

❑ **Here is list of some methods of get object of LocalDate class**

❖ static LocalDate now()

❖ static LocalDate of(int year, int month, int dayOfMonth)

❖ static LocalDate parse(CharSequence text)

❖ static LocalDate parse(CharSequence text, DateTimeFormatter formatter)

The LocalTime class

- A time without a time-zone in the ISO-8601 calendar system, such as 10:15:30.
- LocalTime is an immutable date-time object that represents a time, often viewed as hour-minute-second. Time is represented to nanosecond precision.
- This class does not store or represent a date or time-zone. Instead, it is a description of the local time as seen on a wall clock.
- Here is list of some methods to get object of LocalTime class

  - static LocalTime now()
  - static LocalTime of(int hour, int minute)
  - static LocalTime of(int hour, int minute, int second)
  - static LocalTime of(int hour, int minute, int second, int nanoOfSecond)
  - static LocalTime parse(CharSequence text)
  - static LocalTime parse(CharSequence text, DateTimeFormatter formatter)

The LocalDateTime class

- A date-time without a time-zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30.
- LocalDateTime is an immutable date-time object that represents a date-time, often viewed as year-month-day-hour-minute-second.
- Other date and time fields, such as day-of-year, day-of-week and week-of-year, can also be accessed. Time is represented to nanosecond precision.
- Most of methods are common to LocalDate and LocalTime class

The DateTimeFormatter class

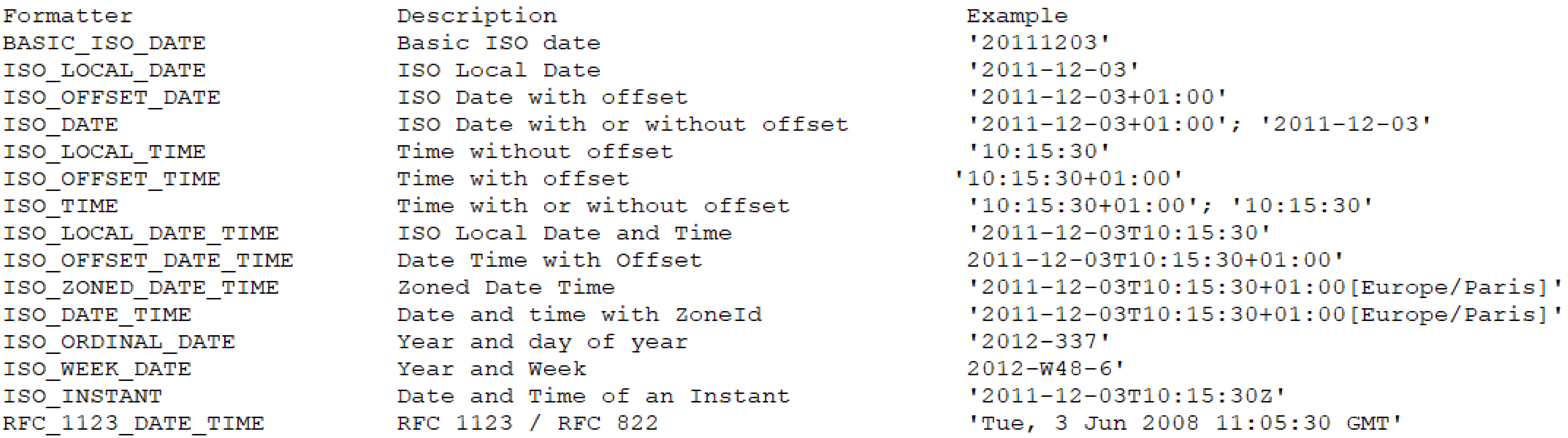
❑ This class of java.time.format package provides the main application entry point for printing and parsing and provides common implementations of DateTimeFormatter:

- ○ Using predefined constants, such as `ISO_LOCAL_DATE`
- ○ Using pattern letters, such as uuuu-MMM-dd
- ○ Using localized styles, such as long or medium

❑ Here is list of some methods to get object of DateTimeFormatter class

❖ static DateTimeFormatter ofPattern(String pattern)
❖ static DateTimeFormatter ofPattern(String pattern, Locale locale)

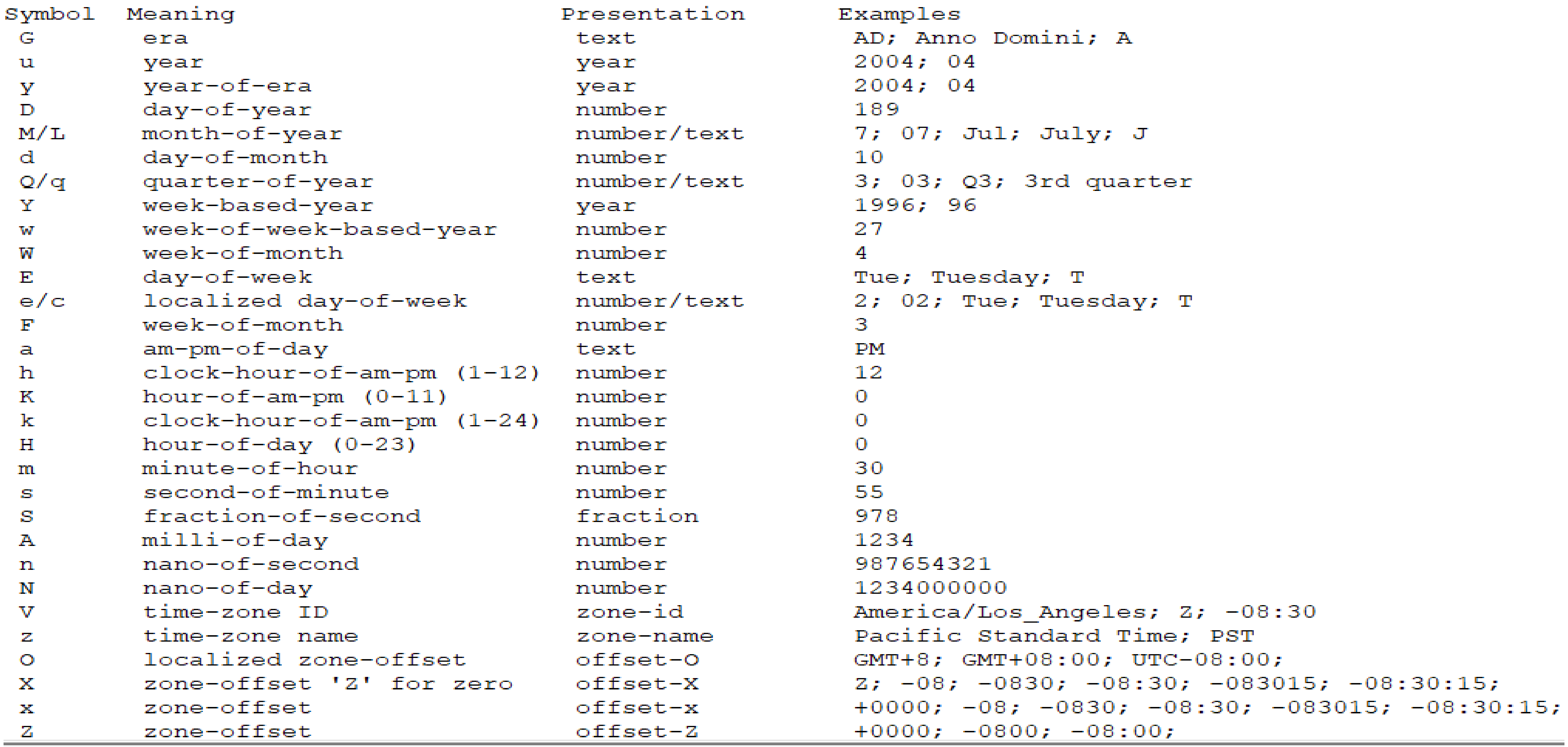
❑ Some predefined formats are



Formatter	Description	Example
BASIC_ISO_DATE	Basic ISO date	'20111203'
ISO_LOCAL_DATE	ISO Local Date	'2011-12-03'
ISO_OFFSET_DATE	ISO Date with offset	'2011-12-03+01:00'
ISO_DATE	ISO Date with or without offset	'2011-12-03+01:00'; '2011-12-03'
ISO_LOCAL_TIME	Time without offset	'10:15:30'
ISO_OFFSET_TIME	Time with offset	'10:15:30+01:00'
ISO_TIME	Time with or without offset	'10:15:30+01:00'; '10:15:30'
ISO_LOCAL_DATE_TIME	ISO Local Date and Time	'2011-12-03T10:15:30'
ISO_OFFSET_DATE_TIME	Date Time with Offset	2011-12-03T10:15:30+01:00'
ISO_ZONED_DATE_TIME	Zoned Date Time	'2011-12-03T10:15:30+01:00[Europe/Paris]'
ISO_DATE_TIME	Date and time with ZoneId	'2011-12-03T10:15:30+01:00[Europe/Paris]'
ISO_ORDINAL_DATE	Year and day of year	'2012-337'
ISO_WEEK_DATE	Year and Week	2012-W48-6'
ISO_INSTANT	Date and Time of an Instant	'2011-12-03T10:15:30Z'
RFC_1123_DATE_TIME	RFC 1123 / RFC 822	'Tue, 3 Jun 2008 11:05:30 GMT'

The DateTimeFormatter class (contd.)

- ❑ **All letters 'A' to 'Z' and 'a' to 'z' are reserved as pattern letters. The following pattern letters are defined**



Symbol	Meaning	Presentation	Examples
G	era	text	AD; Anno Domini; A
u	year	year	2004; 04
y	year-of-era	year	2004; 04
D	day-of-year	number	189
M/L	month-of-year	number/text	7; 07; Jul; July; J
d	day-of-month	number	10
Q/q	quarter-of-year	number/text	3; 03; Q3; 3rd quarter
Y	week-based-year	year	1996; 96
w	week-of-week-based-year	number	27
W	week-of-month	number	4
E	day-of-week	text	Tue; Tuesday; T
e/c	localized day-of-week	number/text	2; 02; Tue; Tuesday; T
F	week-of-month	number	3
a	am-pm-of-day	text	PM
h	clock-hour-of-am-pm (1-12)	number	12
K	hour-of-am-pm (0-11)	number	0
k	clock-hour-of-am-pm (1-24)	number	0
H	hour-of-day (0-23)	number	0
m	minute-of-hour	number	30
s	second-of-minute	number	55
S	fraction-of-second	fraction	978
A	milli-of-day	number	1234
n	nano-of-second	number	987654321
N	nano-of-day	number	1234000000
V	time-zone ID	zone-id	America/Los_Angeles; Z; -08:30
z	time-zone name	zone-name	Pacific Standard Time; PST
O	localized zone-offset	offset-O	GMT+8; GMT+08:00; UTC-08:00;
X	zone-offset 'Z' for zero	offset-X	Z; -08; -0830; -08:30; -083015; -08:30:15;
x	zone-offset	offset-x	+0000; -08; -0830; -08:30; -083015; -08:30:15;
Z	zone-offset	offset-Z	+0000; -0800; -08:00;

The Duration class

- This class models a quantity or amount of time in terms of seconds and nanoseconds. It can be accessed using other duration-based units, such as minutes and hours.
- In addition, the DAYS unit can be used and is treated as exactly equal to 24 hours, thus ignoring daylight savings effects.
- The range of a duration requires the storage of a number larger than a long. To achieve this, the class stores a long representing seconds and an int representing nanosecond-of-second, which will always be between 0 and 999,999,999. The model is of a directed duration, meaning that the duration may be negative.
- This class is immutable and thread-safe.

The Period class

- A date-based amount of time in the ISO-8601 calendar system, such as '2 years, 3 months and 4 days'.
- This class models a quantity or amount of time in terms of years, months and days.
- The Period class is useful for daylight saving time

The ChronoUnit Enum

- This ChronoUnit enum belongs to java.time.temporal package. it is the most useful structure in java.time API.
- ChronoUnit is an enum that implements the TemporalUnit interface, which provides the standard units used in the Java Date Time API.
- With the help of this enum also we can get the difference between 2 dates in more precise way.

Why do need Collection Framework

Limitation with the Array

- ❑ Arrays are having fixed size in nature
- ❑ No predefined methods for searching and sorting
- ❑ To keep distinct elements only, to keep the elements sorted always despite of several insertions and deletions then developer has to write enormous amount of code and writing highly efficient and error free code require a lot of skill & experience.

Solution: Collection framework in java

What is collection?

- ❑ A collection is simply an object which has group multiple elements into a single unit.
- ❑ Collections are used to store retrieve, manipulate, and communicate aggregate data.
- ❑ Collection framework contains
  1. Interfaces
  2. Implementations
  3. Algorithms
- ❑ Tip: All classes and interfaces related to the collection framework are in the java.util package.

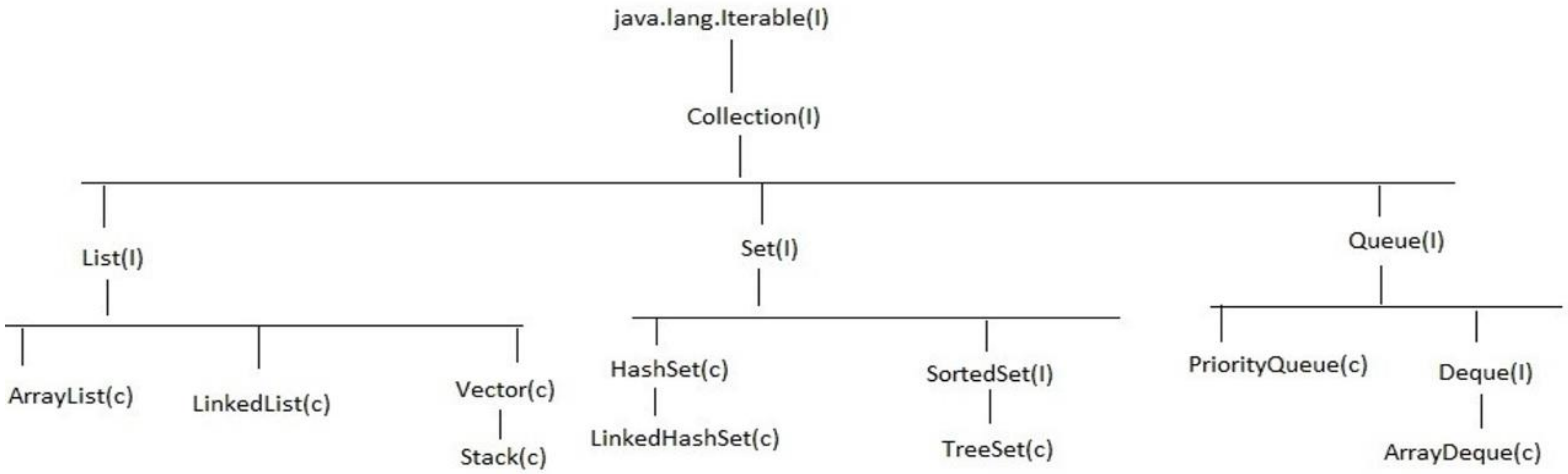
What collection framework provides?

- Collection framework provides a unified approach for representing and manipulating collections. Collection framework is designed to meet following goals
- Provides high performance implementation of most widely used collection object. It provides implementation of dynamic arrays, linked list and hash tables hence programmer does not need to develop their own code to implement these.
- Provides high degree of interoperability and work in similar manner for different type of collections.
- Designed with keeping flexibility in mind, It is very easy to extend and adapt inbuilt interface.

Collection framework contains two sections:-

- Normal Collection (multiple objects are handled individually)
- Map (multiple objects are handled in the form of pair (key-value)).

- Let us start with some discussion about normal Collection.



java.lang.Iterable(I)
Collection(I)
List(I)
Set(I)
Queue(I)
ArrayList(c)
LinkedList(c)
Vector(c)
Stack(c)
HashSet(c)
LinkedHashSet(c)
SortedSet(I)
TreeSet(c)
PriorityQueue(c)
Deque(I)
ArrayDeque(c)