Generics

08 December 2021 09:28

Generic Form: It is a recurring code pattern implemented in a strictly typed language so that it can be reused with different data-types in a type-safe manner.

Java (version 5.0 onwards) provides syntactical support for implementing generic forms with reference types by enabling its compiler to identify matching object types and to automatically perform explicit type conversions. In Java a generic declaration contains at least one type argument with following characteristics

1. It stands for java.lang.Object and as such it can be substituted by different reference types but it only supports members of java.lang.Object by default (type erasure).
2. It can be bounded (using *extends* statement) by a reference type R so that it additionally supports members of :R but it can only be substituted by types which support implicit conversion to (inherit from) R.

Wild-Card Substitution - For a generic type G<T> is invariant over its type argument T i.e G<U> cannot be substituted by G<V> even if V is a sub-type of U. G<T> can appear in a declaration as

1. G<? extends U> which can be substituted by any G<V> if V supports implicit conversion to U but only members of G in which T does not appear as parameter type can applied to this declaration. Also G<?> is short for G<? extends Object>
2. G<? super U> which can be substituted by any G<V> if V supports implicit conversion from U but only members of G in which T does not appear as return type can applied to this declaration.

Generic Collection: It is a container of multiple elements of any given type which exposes a common interface for supporting addition and iteration of those elements. There are two main types of generic collections:

1. **List** - it arranges indexed elements in a sequential manner so that on iteration it can yield them in the order of the time of their addition.
2. **Set** - it arranges unique elements in a manner associated with their behavior so that on iteration it can yield them in the order indicated by their states. A *map* which is also known as a *dictionary*

contains a set whose each element is composed of a unique key and a value identified by that key.

Java runtime library extends its java.lang.*Iterable*<E> interface to define java.util.*Collection*<E> interface which serves as a common interface for all generic collections in Java. It further extends this interface to define
(a) java.util.*List*<V> interface which is implemented by list collections such as java.util.ArrayList<V> and java.util.LinkedList<V>
(b) java.util.*Set*<K> interface which is implemented by set collections such as java.util.HashSet<K> and java.util.TreeSet<K>.
The library also includes java.util.HashMap<K,V> and java.util.TreeMap<K,V> both of which implement java.util.*Map*<K,V> interface and respectively contain java.util.HashSet<P> and java.utilTreeSet<P>

Characteristic	ArrayList	LinkedList	HashSet(Map)	TreeSet(Map)
ref/entry	1	3	1 (2)	3 (4)
adding	O(1)	O(1)	O(n)	O(log n)
searching	O(n)	O(n)	O(n)	O(log n)
indexing	O(1)	O(n)	n/a	n/a