Generics

16 April 2022 09:30

Java Generics: It is syntactical support offered by Java language for implementing common code-patterns which can be reused with different reference types in a type-safe manner. It allows the compiler to identify matching types in a declaration and to automatically perform type conversions which are otherwise explicit. A generic declaration contains at least one type parameter with following characteristics

1. It can be substituted by any reference type by default since it is treated as java.lang.Object (erasure) by the compiler but then it only supports members defined in java.lang.Object.
2. It can be bounded (using extends statement) by a known reference type so that it also supports members of that bounding type but then it can only be substituted by a type which supports implicit conversion (inherits from) to that bounding type.

Wild-Card Substitution: A generic type G<T> is invariant over its type-parameter T i.e G<U> cannot be substituted by G<V> irrespective of relationship between U and V. However a variant form of G<T> can used for declaration as

1. G<? extends U> which can be substituted by G<V> if V = U or V supports implicit conversion *to* U but members of G in which T appears as an *argument* type cannot be applied to this declaration.
2. G<? super U> which can be substituted by G<V> if V = U or V supports implicit conversion *from* U but members of G in which T appears as a *return* type cannot be applied to this declaration.

Generic Collection: It is an object of a generic class which groups multiple elements of a given type and provides access to these elements in a type-safe manner. The *java.util* package of Java runtime library defines *Collection*<E> interface which extends *java.lang.Iterable*<E> interface to specify standard support for adding/removing elements from/to a collection. This interface has following sub-interfaces in its package

1. ***List*<E>** which is implemented by a collection which allows its elements to be accessed using their sequential indexes. The java.util package includes ArrayList<E> (less memory, fast indexing) and LinkedList<E> (more memory, slow indexing) which implement this interface.

2. ***Set*<E>** which implemented by a collection which only allows non-duplicate elements whose identities are inferred from their behavior. The java.util package includes HashSet<E> (less memory, slow operations) and TreeSet<E> (more memory, fast operation) which implement this interface.
   A set whose each element contains a unique key and a value associated with that key is called a map. The java.util package includes HahMap<K, V> (less memory, slow operations) and TreeMap<K, V> (more memory, fast operations) which implement its Map<K, V> interface.