JAVA is a major feature release of JAVA programming language development. Its initial version was released on 18 March 2014. With the Java 8 release, Java provided supports for functional programming, new JavaScript engine, new APIs for date time manipulation, new streaming API, etc.

New Features

- **Lambda expression** − Adds functional processing capability to Java.
- **Method references** − Referencing functions by their names instead of invoking them directly. Using functions as parameter.
- **Default method** − Interface to have default method implementation.
- **New tools** − New compiler tools and utilities are added like ‘jdeps’ to figure out dependencies.
- **Stream API** − New stream API to facilitate pipeline processing.
- **Date Time API** − Improved date time API.
- **Optional** − Emphasis on best practices to handle null values properly.
- **Nashorn, JavaScript Engine** − A Java-based engine to execute JavaScript code.

Consider the following code snippet.

Live Demo
```
import java.util.Collections;
import java.util.List;
import java.util.ArrayList;
import java.util.Comparator;

public class Java8Tester {

   public static void main(String args[]) {

      List<String> names1 = new ArrayList<String>();
      names1.add("Mahesh ");
      names1.add("Suresh ");
      names1.add("Ramesh ");
      names1.add("Naresh ");
      names1.add("Kalpesh ");

      List<String> names2 = new ArrayList<String>();
      names2.add("Mahesh ");
      names2.add("Suresh ");
      names2.add("Ramesh ");
      names2.add("Naresh ");
      names2.add("Kalpesh ");

      Java8Tester tester = new Java8Tester();
      System.out.println("Sort using Java 7 syntax: ");

      tester.sortUsingJava7(names1);
      System.out.println(names1);
      System.out.println("Sort using Java 8 syntax: ");
      tester.sortUsingJava8(names2);
      System.out.println(names2);
   }

   //sort using java 7
   private void sortUsingJava7(List<String> names) {
      Collections.sort(names, new Comparator<String>() {
         @Override
         public int compare(String s1, String s2) {
            return s1.compareTo(s2);
         }
      });
   }

   //sort using java 8
   private void sortUsingJava8(List<String> names) {
      Collections.sort(names, (s1, s2) -> s1.compareTo(s2));
   }
}
```

Run the program to get the following result.

```
Sort using Java 7 syntax:
[ Kalpesh Mahesh Naresh Ramesh Suresh ]
Sort using Java 8 syntax:
[ Kalpesh Mahesh Naresh Ramesh Suresh ]
```

Here the **sortUsingJava8()** method uses sort function with a lambda expression as parameter to get the sorting criteria.