

# Difference between Callable and Runnable in Java - Thread Interview question

The difference between the Callable and Runnable interface in Java is one of the interesting questions from my list of [Top 15 Java multi-threading questions](#), and it's also very popular in various Java Interviews. The Callable interface is newer than the Runnable interface and was added on Java 5 release along with other major changes e.g. [Generics](#), [Enum](#), [Static imports](#), and [variable argument method](#). Though both Callable and Runnable interfaces are designed to represent a task, which can be executed by any thread, there is some significant difference between them.

In my opinion, the major difference between the Callable and Runnable interface is that Callable can return the result of an operation performed inside the `call()` method, which was one of the limitations of the Runnable interface.

Another significant difference between the Runnable and Callable interfaces is the ability to throw [checked exceptions](#). The Callable interface can throw [checked exceptions](#) because its `call` method throws Exceptions.

By the way, sometimes this question is also asked as a follow-up question of another classic [difference between Runnable and Thread in Java](#). Commonly `FutureTask` is used along with Callable to get the result of asynchronous computation task performed in `call()` method.

## **Callable vs Runnable interface in Java**

As I explained the major differences between a Callable and Runnable interface in the last section. Sometimes this question is also asked as the difference between call() and run() method in Java. All the points discussed here are equally related to that question as well. Let's see them in point format for better understanding :

- 1) The Runnable interface is older than Callable which is there from JDK 1.0, while Callable is added on Java 5.0.
- 2) Runnable interface has run() method to define task while Callable interface uses call() method for task definition.
- 3) run() method does not return any value, its return type is void while the call method returns a value. The Callable interface is a **generic parameterized interface** and a Type of value is provided when an instance of Callable implementation is created.
- 4) Another difference in the run and call method is that the run method can not **throw** checked exceptions while the call method can throw checked exceptions in Java.

Here is a nice summary of all the **differences between Callable and Runnable in Java**:

## Callable

- **Runnable**
  - “run” method runs in background. No return values, but run can do side effects.
  - Use “execute” to put in task queue
- **Callable**
  - “call” method runs in background. It returns a value that can be retrieved after termination with “get”.
  - Use “submit” to put in task queue.
  - Use `invokeAny` and `invokeAll` to block until value or values are available
    - Example: you have a list of links from a Web page and want to check status (404 vs. good). Submit them to a task queue to run concurrently, then `invokeAll` will let you see return values when all links are done being checked.

That's all on the difference between Callable and Runnable interface in Java or difference between `call()` and `run()` method.