# Java 9 Try With Resources Improvements

pramodbablad September 20, 2021 0

Try with resources blocks are introduced from Java 7. In these blocks, resources used in try blocks are auto-closed. No need to close the resources explicitly. But, Java 7 try with resources has one drawback. It requires resources to be declared locally within try block. It doesn't recognize resources declared outside the try block. That issue has been resolved in Java 9. In this post, we will see how the resources are closed before Java 7, how the resources are closed after the introduction of try with resources blocks from Java 7 and improvements made to try with resources in Java 9.

## How the resources are closed before Java 7?

Any resource (File or database connection or network connection etc…) needs to be released after they are used to avoid resource leaks and also make them available for others to use. Before Java 7, try with finally blocks are used to close the resources. As you know, finally blocks are executed irrespective of whether try block is successfully executed or not. This makes sure that resources are released after their usage in try block if you keep resources closing statements in finally block.

For example, in the below program, `FileOutputStream fos` is the resource which is used in try block to write into `Resource.txt` and closed in finally block.

```
import java.io.FileNotF
import java.io.FileOutp
import java.io.IOExcept

```

```
public class Resourcesh
{
    public static void
    {
        FileOutputStre

        try
        {
            //Using the

            fos.write('
        }
        catch (IOExcept
        {
            e.printStac
        }
        finally
        {
            //Releasing

            try
            {
                fos.clo
            }
            catch (IOEx
            {
                e.print
            }
        }
    }
}
```

## How the resources are closed after Java 7?

With the introduction of try with resources in Java 7, closing the resources have become even easier. There is no need to explicitly close the resources as in the above example. Try with resources auto closes the resources used in try block.

The above program using Java 7 try-with resources can be written as follows.

```
import java.io.FileNotF
import java.io.FileOutp
import java.io.IOExcept

public class Resourcesh
{
    public static void
    {
        FileOutputStre

        try(FileOutputS
        {
            //Using the

            fos.write('
        }
        catch (IOExcept
        {
            e.printStac
        }

        //No need to cl
        //Resources are
    }
}
```

Notice that resources used in try block are implicitly closed. There is no need to close them explicitly.

**Drawback of Java 7 Try-With-Resources :**

One drawback of Java 7 try with resources is that resources need to be declared within () of try block or else need to assign reference of resource declared outside to local variable of try block as in the above example. It doesn't recognize resources declared outside its body. This issue has been addressed in Java 9.

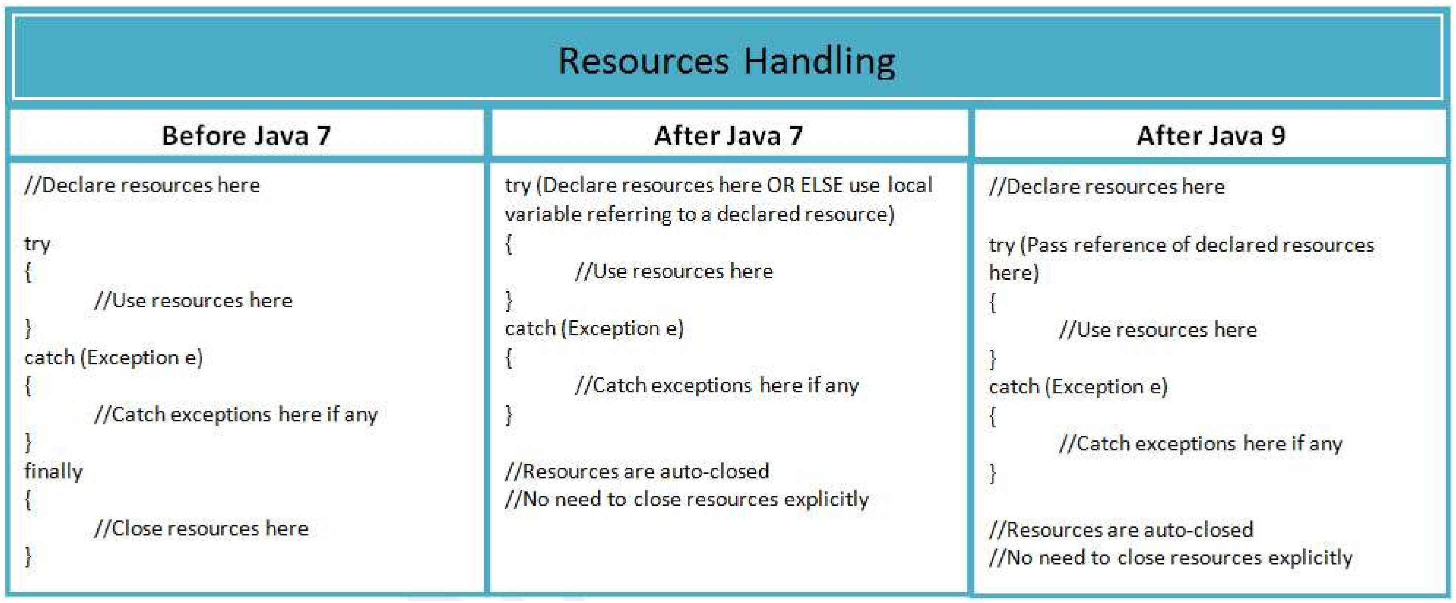
## Java 9 Try With Resources Improvements :

From Java 9, try with resources will recognize resources declared outside its body. You can pass the reference of resource declared outside directly to try block. There is no need to declare resources locally within try block.

From Java 9, try-with-resources can be written as follows.

```
import java.io.FileNotF
import java.io.FileOutp
import java.io.IOExcept

public class Java9TryWi
{
    public static void
    {
        FileOutputStrea

        try(fos)     //
        {
            //Using the

            fos.write('
        }
        catch (IOExcept
        {
            e.printStac
        }

        //No need to cl
        //Resources are
    }
}
```

Below table shows how resources can be handled before Java 7, after Java 7 and after Java 9.



| Resources Handling | | |
|---|---|---|
| **Before Java 7** | **After Java 7** | **After Java 9** |
| //Declare resources here<br><br>try<br>{<br>    //Use resources here<br>}<br>catch (Exception e)<br>{<br>    //Catch exceptions here if any<br>}<br>finally<br>{<br>    //Close resources here<br>} | try (Declare resources here OR ELSE use local variable referring to a declared resource)<br>{<br>    //Use resources here<br>}<br>catch (Exception e)<br>{<br>    //Catch exceptions here if any<br>}<br><br>//Resources are auto-closed<br>//No need to close resources explicitly | //Declare resources here<br><br>try (Pass reference of declared resources here)<br>{<br>    //Use resources here<br>}<br>catch (Exception e)<br>{<br>    //Catch exceptions here if any<br>}<br><br>//Resources are auto-closed<br>//No need to close resources explicitly |

**Also Read :**