Java try-with-resources

Support for *try-with-resources* — introduced in Java 7 — allows us to declare resources to be used in a *try* block with the assurance that the resources will be closed after the execution of that block.The resources declared need to implement the *AutoCloseable* interface.

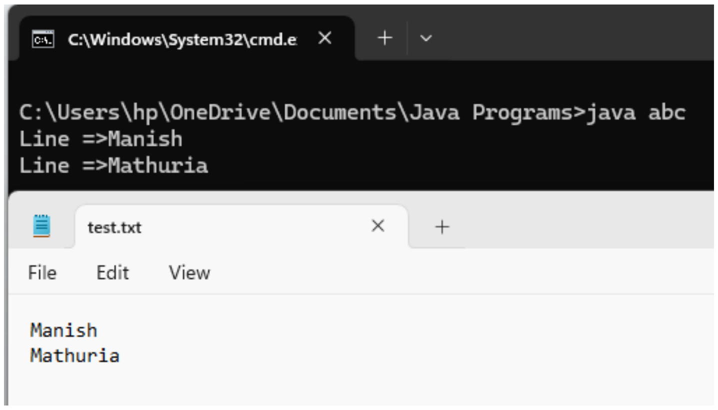
The try-with-resources statement automatically closes all the resources at the end of the statement. A resource is an object to be closed at the end of the program.

Its syntax is:

```
try (resource declaration)
{
  // use of the resource
}
catch (ExceptionType e1)
{
  // catch block
}
```

Example:

```
import java.io.*;
class abc
{
        public static void main(String[] args)
        {
                String line;
                try(BufferedReader br = new BufferedReader(new FileReader("test.txt")))
                {
                        while ((line = br.readLine()) != null)
                        {
                        System.out.println("Line =>"+line);
                        }
                }
                catch(IOException e)
                {
                        System.out.println("IOException in try block =>" + e.getMessage());
                }
        }
}
```



```
C:\Users\hp\OneDrive\Documents\Java Programs>java abc
Line =>Manish
Line =>Mathuria
```

```
Manish
Mathuria
```

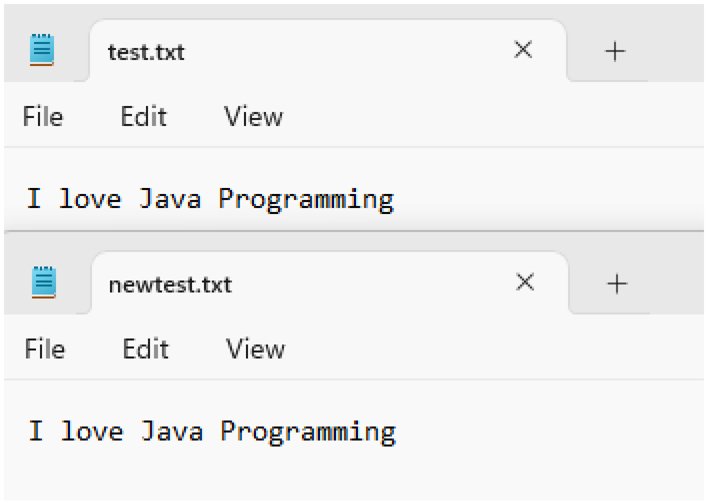
Advantages of using try-with-resources-

1. **Finally block not required to close the resource. Example:**

```
import java.io.*;
class abc
{
	public static void main(String[] args)
	{
		BufferedReader br = null;
		String line;
		try
		{
			System.out.println("Entering try block");
			br = new BufferedReader(new FileReader("test.txt"));
			while ((line = br.readLine()) != null)
			{
			System.out.println("Line =>"+line);
			}
		}
		catch(IOException e)
		{
			System.out.println("IOException in try block =>" + e.getMessage());
		}
		finally
		{
			System.out.println("Entering finally block");
			try
			{
			if (br != null)
				{
				br.close();
			}
			}
			catch(IOException e)
			{
			System.out.println("IOException in finally block =>"+e.getMessage());
			}
		}
	}
}
```

2. try-with-resources with multiple resources. Example:

```
import java.io.*;
import java.util.*;
class abc
{
        public static void main(String[] args) throws IOException
        {
                try (Scanner scanner = new Scanner(new File("test.txt"));
                PrintWriter writer = new PrintWriter(new File("newtest.txt")))
                {
                        while (scanner.hasNext())
                        {
                        writer.print(scanner.nextLine());
                        }
                }
        }
}
```



```
I love Java Programming
```

```
I love Java Programming
```