# Read a text file

**Estimated time needed:** 15 minutes

In this lab, you will learn how to read content of a text file and print it out.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully online, integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

## Learning Objectives

After completing this lab, you will be able to:

- Understand how to open a file a text file
- Know how to read the contents of text file using java.io packages
- Handle the checked exceptions that occur during file I/O (Input/Output).
- Print the contents of the file onto console

# Reading a File - Using Scanner

1. Create a project directory by running the following command.

   ```
   mkdir my_fileio_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_fileio_proj/src
   mkdir -p my_fileio_proj/classes
   mkdir -p my_fileio_proj/test
   cd my_fileio_proj
   ```

3. Now create a file named `ReadFileExampleWithScanner.java` inside the src directory.

```
touch /home/project/my_fileio_proj/src/ReadFileExampleWithScanner.java
```

4. Click the following button to open the file for editing.

Open **ReadFileExampleWithScanner.java** in IDE

5. Copy and paste the code in `ReadFileExampleWithScanner.java`. Read and the comments carefully to understand what the code does. See how the file validity is checked using exception handling and how FileNotFoundException is handled.

```
// Import necessary classes for file reading and user input
import java.io.FileReader;
import java.util.Scanner;
import java.io.FileNotFoundException;
// Define the main class
class ReadFileExampleWithScanner {
    public static void main(String[] args) {
        // Specify the file path
        try {
            // Create a Scanner object to read user input from the console
            Scanner scanner = new Scanner(System.in);
            // Prompt the user to enter the name of the file they want to read
            System.out.println("Enter the name of the file you want to read.");
            // Create a new Scanner object to read from the file specified by the user
            // FileReader is used to read the file, and the file name is obtained from the user input
            Scanner fileScanner = new Scanner(new FileReader(scanner.nextLine()));
            // Loop through the file line by line
            while(fileScanner.hasNext()) {
                // Read the next line from the file
                String fileLine = fileScanner.nextLine();
                // Print the line to the console
                System.out.println(fileLine);
            }
            // Close the file scanner to free up resources (optional but recommended)
            fileScanner.close();
        } catch (FileNotFoundException e) {
            // Handle the case where the file is not found
            System.err.println("Error reading file: " + e.getMessage());
        }
    }
}
```

This program reads and displays the contents of a file specified by the user. It prompts the user to enter the name of the file they want to read. It opens the specified file using FileReader and reads its contents line by line using a Scanner. Each line of the file is printed to the console. If the file is not found, the program catches the FileNotFoundException and prints an error message.

6. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/ReadFileExampleWithScanner.java
```

7. Set the `CLASSPATH` variable.

```
export CLASSPATH=$CLASSPATH:/home/project/my_fileio_proj/classes
```

8. Run the program and test with variable combinations.

```
java ReadFileExampleWithScanner
```

9. When prompted for an input provide any file name in the current directory or as entire path.

For example,

```
/home/project/my_fileio_proj/src/ReadFileExampleWithScanner.java
```

## Sample output when reading is successful, provided a file exists.

```
Lav: +1(201)920-2243
Sam: +61498456533
```

### Sample output with error-handling

```
Enter the name of the file you want to read.
somefile.txt
Error reading file: somefile (No such file or directory)
```

# Reading a file - Using Files class

Now that you have added, updated, and deleted entries in this section you will sort the HashMap by name and store the content to a file.

1. Create a file named `ReadFileExample.java` inside the src directory.

   ```
   touch /home/project/my_fileio_proj/src/ReadFileExample.java
   ```

2. Click the following button to open the file for editing.

Open **ReadFileExample.java** in IDE

3. Copy and paste the code in `ReadFileExample.java`. Read and the comments carefully to understand what the code does. See how the file validity is checked using exception handling and how IOException is handled.

```
// Import necessary classes for file operations and user input
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Scanner;
// Define the main class
public class ReadFileExample {
    public static void main(String[] args) {
        // Create a Scanner object to read user input from the console
        Scanner scanner = new Scanner(System.in);
        // Prompt the user to enter the name of the file they want to read
```

```
        System.out.println("Enter the name of the file you want to read.");
        // Get the file path from the user input and convert it to a Path object
        Path filePath = Paths.get(scanner.nextLine());
        try {
            // Read the entire content of the file as a single string
            String content = Files.readString(filePath);
            // Print the file content to the console
            System.out.println(content);
        } catch (IOException e) {
            // Handle the case where an I/O error occurs (e.g., file not found or cannot be read)
            System.err.println("Error reading file: " + e.getMessage());
        }
    }
}
```

4. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/ReadFileExample.java
```

4. Run the program and test with variable combinations.

```
java ReadFileExample
```

## Sample output when reading is successful

```
Enter the name of the file you want to read.
taskBook.txt
homework: homework
finish MAth homework by 5.00 PM before basketball game
3
New
laundry: laundry
Get the laundry done
2
Completed
```

### Sample output with error-handling

```
Enter the name of the file you want to read.
somefile
Error reading file: somefile
```

# Practice Exercise

1. Create a java program where you will get names of file in a loop as long as the users wants to enter and read and display the contents.

▶ Click here for sample code

# Conclusion

In this lab, you learned how to read content from a file using two different approaches.

## Author(s)

Lavanya