Chapter Six

Java File Handling

MULUGETA G. and Sufian K.

Java File

- File handling is an important part of any application.
- Java has several methods for creating, reading, updating, and deleting files.
- The File class from the java.io package, allows us to work with files.
- To use the File class, create an object of the class, and specify the filename or directory name:

```
import java.io.File;  // Import the File class

File myObj = new File("filename.txt"); // Specify the filename
```

Methods

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

Create a File

- To create a file in Java, you can use the createNewFile() method.
- This method returns a boolean value:
  - true if the file was successfully created, and
  - false if the file already exists.
- Note that the method is enclosed in a try...catch block.
- This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason):

Example

```
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle errors

public class CreateFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
      } else {
        System.out.println("File already exists.");
      }   }
catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }  } }
```

- To create a file in a specific directory (requires permission), specify the path of the file and use double backslashes to escape the "\" character (for Windows).
- On Mac and Linux you can just write the path, like: /Users/name/filename.txt
- Example:

```
File myObj = new
File("C:\\Users\\MyName\\filename.txt");
```

Write to a File

- Let's use the FileWriter class together with its write() method to write some text to the file we created.
- Note that when you are done writing to the file, you should close it with the close() method:

Example

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors
public class WriteToFile {
  public static void main(String[] args) {
    try {
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Files in Java might be tricky, but it is fun
enough!");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    } } }
```

Read a File

- we can use the Scanner class to read the contents of the text file we created.

```
public class ReadFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    } } }
```

Get File Information

- To get more information about a file, use any of the File methods:

```
public class GetFileInfo {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.exists()) {
      System.out.println("File name: " + myObj.getName());
      System.out.println("Absolute path: " +
myObj.getAbsolutePath());
      System.out.println("Writeable: " + myObj.canWrite());
      System.out.println("Readable " + myObj.canRead());
      System.out.println("File size in bytes " + myObj.length());
    } else {
      System.out.println("The file does not exist.");
    }}}
```

Delete a File

- To delete a file in Java, use the delete() method:

```
public class DeleteFile {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.delete()) {
      System.out.println("Deleted the file: " +
myObj.getName());
    } else {
      System.out.println("Failed to delete the file.");
    } }}
```

Thank You