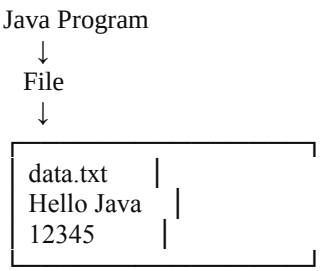
File Handling in Java

File handling in Java means **creating, reading, writing, updating, and deleting files** using Java programs.

Java provides several classes for file handling, mainly in the java.io package and also java.nio.file.

Think of it like:



1. Creating a File

Java provides the File class for basic file operations.

```
import java.io.File;
import java.io.IOException;

public class Main {
   public static void main(String[] args) throws IOException {

      File file = new File("data.txt");

      if (file.createNewFile()) {
         System.out.println("File created");
      } else {
         System.out.println("File already exists");
      }
   }
}
```

createNewFile() creates the file if it doesn't already exist.

---

2. Writing to a File

We can use FileWriter to write text into a file.

```
import java.io.FileWriter;
import java.io.IOException;

public class Main {
   public static void main(String[] args) throws IOException {

      FileWriter writer = new FileWriter("data.txt");

      writer.write("Hello Java!");
      writer.write("\nWelcome to File Handling.");

      writer.close();
   }
}
```

The file will contain:

```
Hello Java!
Welcome to File Handling.
```

Important

Always close the writer:

`writer.close();`

Otherwise, data may not be properly saved and system resources remain occupied.

---

3. Reading from a File

One simple way is using `FileReader`.

```
import java.io.FileReader;
import java.io.IOException;

public class Main {
   public static void main(String[] args) throws IOException {

      FileReader reader = new FileReader("data.txt");

      int ch;

      while ((ch = reader.read()) != -1) {
         System.out.print((char) ch);
      }

      reader.close();
   }
}
```

How does `read()` work?

`reader.read()`

reads one character at a time.

When there are no more characters:

`-1`

is returned.

---

4. Reading Using `BufferedReader`

For reading text more conveniently, we can use `BufferedReader`.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Main {
   public static void main(String[] args) throws IOException {

      BufferedReader reader =
            new BufferedReader(new FileReader("data.txt"));

      String line;

      while ((line = reader.readLine()) != null) {
         System.out.println(line);
      }

      reader.close();
   }
}
```

`readLine()` reads the file **one line at a time**.

---

5. Appending Data to a File

Normally:

```
FileWriter writer = new FileWriter("data.txt");
```

can overwrite the existing contents.

To append instead:

```
FileWriter writer = new FileWriter("data.txt", true);

writer.write("\nThis is new data.");

writer.close();
```

The `true` means **append mode**.

---

6. Deleting a File

```
import java.io.File;

public class Main {
   public static void main(String[] args) {

      File file = new File("data.txt");

      if (file.delete()) {
         System.out.println("File deleted");
      } else {
         System.out.println("File not found");
      }
   }
}
```

---

7. Checking File Information

The `File` class can provide information about a file.

```
File file = new File("data.txt");

System.out.println(file.exists());
System.out.println(file.getName());
System.out.println(file.getAbsolutePath());
System.out.println(file.length());
```

Some useful methods:

Method	Purpose
exists()	Checks whether file exists
createNewFile()	Creates a new file
delete()	Deletes a file
getName()	Gets file name
getAbsolutePath()	Gets complete path
length()	Gets file size in bytes
isFile()	Checks whether it is a file
isDirectory()	Checks whether it is a directory

---

8. try-catch and File Handling

File operations can cause exceptions.

For example, the file might not exist.

So we can handle exceptions:

```
import java.io.FileReader;
import java.io.IOException;

public class Main {
   public static void main(String[] args) {

      try {
         FileReader reader = new FileReader("data.txt");

         int ch;

         while ((ch = reader.read()) != -1) {
            System.out.print((char) ch);
         }

         reader.close();

      } catch (IOException e) {
         System.out.println("An error occurred.");
      }
   }
}
```

IOException is a common exception in file handling.

---

9. Modern File Handling with Files

Modern Java also provides the java.nio.file package.

For example, reading an entire file:

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.io.IOException;

public class Main {
   public static void main(String[] args) throws IOException {

      String data = Files.readString(Path.of("data.txt"));

      System.out.println(data);
   }
}
```

Writing:

```
Files.writeString(
   Path.of("data.txt"),
   "Hello Java"
);
```

This approach is often simpler for basic operations.