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):

```
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();

    }

  }

}
```

Write To a File

In the following example, we use the `FileWriter` class together with its `write()` method to write some text to the file we created in the example above. Note that when you are done writing to the file, you should close it with the `close()` method:

```
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

In the previous chapter, you learned how to create and write to a file.

In the following example, we use the `Scanner` class to read the contents of the text file we created in the previous chapter:

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

import java.io.FileNotFoundException;  // Import this class to
handle errors

import java.util.Scanner; // Import the Scanner class to read text
files


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:

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


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.");
    }
  }
}
```

Note: There are many available classes in the Java API that can be used to read and write files in Java: `FileReader`, `BufferedReader`, `Files`, `Scanner`, `FileInputStream`, `FileWriter`, `BufferedWriter`, `FileOutputStream`, etc. Which one to use depends on the Java version you're working with and whether you need to read bytes or characters, and the size of the file/lines etc.

Delete a File

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

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


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.");
    }
  }
}
```

Delete a Folder

You can also delete a folder. However, it must be empty:

```
import java.io.File;


public class DeleteFolder {
  public static void main(String[] args) {
    File myObj = new File("C:\\Users\\MyName\\Test");
    if (myObj.delete()) {
      System.out.println("Deleted the folder: " + myObj.getName());
    } else {
      System.out.println("Failed to delete the folder.");
    }
  }
}
```

Reverse a String

You can easily reverse a string by characters with the following example:

```
String originalStr = "Hello";

String reversedStr = "";
```

```
for (int i = 0; i < originalStr.length(); i++) {
  reversedStr = originalStr.charAt(i) + reversedStr;
}

System.out.println("Reversed string: "+ reversedStr);
```