1. Create and Write to a File: Write a Java program that prompts the user for a diary entry, then creates a file named "diary.txt" and writes the current date followed by the user's entry into this file. Ensure the program checks if the file already exists and informs the user, to avoid overwriting any previous content.

2. Read from a File: Write a Java application that opens the "diary.txt" file created in the previous question and displays its content on the console. The program should handle cases where the file does not exist by displaying an appropriate error message.

3. Append Content to an Existing File: Write a Java program that adds a new diary entry to the "diary.txt" file without overwriting its existing content. The program should ask the user for the new entry and append it to the file along with a timestamp.

4. List Files and Directories: Write a program in Java that asks the user for a directory path and then lists all files and subdirectories in that directory. If the directory does not exist, the program should inform the user.

5. Filter and List Specific File Types: Create a Java application that lists all the ".txt" files in a given directory. The program should prompt the user for the directory path and then display a list of all text files found in that directory.

6. Delete a Specific File: Write a Java program where the user can enter the name of a file to be deleted from the system. The program should check if the file exists and delete it, providing a confirmation message upon successful deletion or an error message if the file does not exist.

```
f.delete()
```

7. Copy File Content: Write a Java program that copies the content from one file (source) to another (destination). The program should prompt the user for both source and destination file paths and perform the copy operation, ensuring that it doesn't overwrite an existing file without user confirmation.

```
File cur=new File(curfile);
File mod=new File(renamefile);
boolean flag=cur.renameTo(mod);
```

8. Rename a File: Develop a Java application that renames a specified file. The program should request the current file name and the new file name from the user, renaming the file accordingly and confirming the action upon completion.

9. Display File Metadata: Create a Java program that displays metadata of a specified file. The user should be able to input the file name, and the program should output the file size, last modified date, and other available attributes.

10. Recursive Directory Listing: Write a Java program that recursively lists all files and subdirectories within a given directory. The program should prompt the user for the directory path and then display a structured list of all contents, including files and directories nested within any subdirectories.

```
System.out.println("*************************");
        System.out.println("* Display File metadata *");
        System.out.println("*************************");
        System.out.println("1. File name: " + file.getName());
        System.out.println("2. Absolute path: " + file.getAbsolutePath
());
        System.out.println("3. File size: " + file.length() + " bytes");
        System.out.println("4. Last modified date: " + formatDate(new
Date(file.lastModified())));
        System.out.print("5. Is directory: ");
```