

File IO & Exception Handling

Lab 9

TA : Hyuna Seo, Kichang Yang, Minkyung Jeong, Jaeyong Kim



SEOUL NATIONAL UNIVERSITY

Announcement

- You should finish the lab practice and submit your job to eTL before the next lab class starts(**Wednesday, 7:00 PM**).
- The answer of the practice will be uploaded after the due.

Overview

- Recap: File IO
- Recap: Exception Handling Review
- Problem 1: Simple Diary Application (ver 2)
- Problem 2: Exceptions Practice

Recap: **FileWriter** Class

- **FileWriter** is used to write characters to a file.
- **FileWriter** creates files to
 - Write (overwrite)
 - Append (add to existing file)


Recap: **FileWriter** Example

```
public void writeLines(String fName, List<String> l) {  
  
    FileWriter fileWriter = new FileWriter(fName);  
    for (String string : l) {  
        fileWriter.write(string);  
    }  
    fileWriter.close();  
  
}
```

Recap: **FileWriter** Example

```
public void writeLines(String fName, List<String> l) {  
  
    FileWriter fileWriter = new FileWriter(fName);  
    for (String string : l) {  
        fileWriter.write(string);  
    }  
    fileWriter.close();  
  
}
```

Unhandled Exception: java.io.IOException

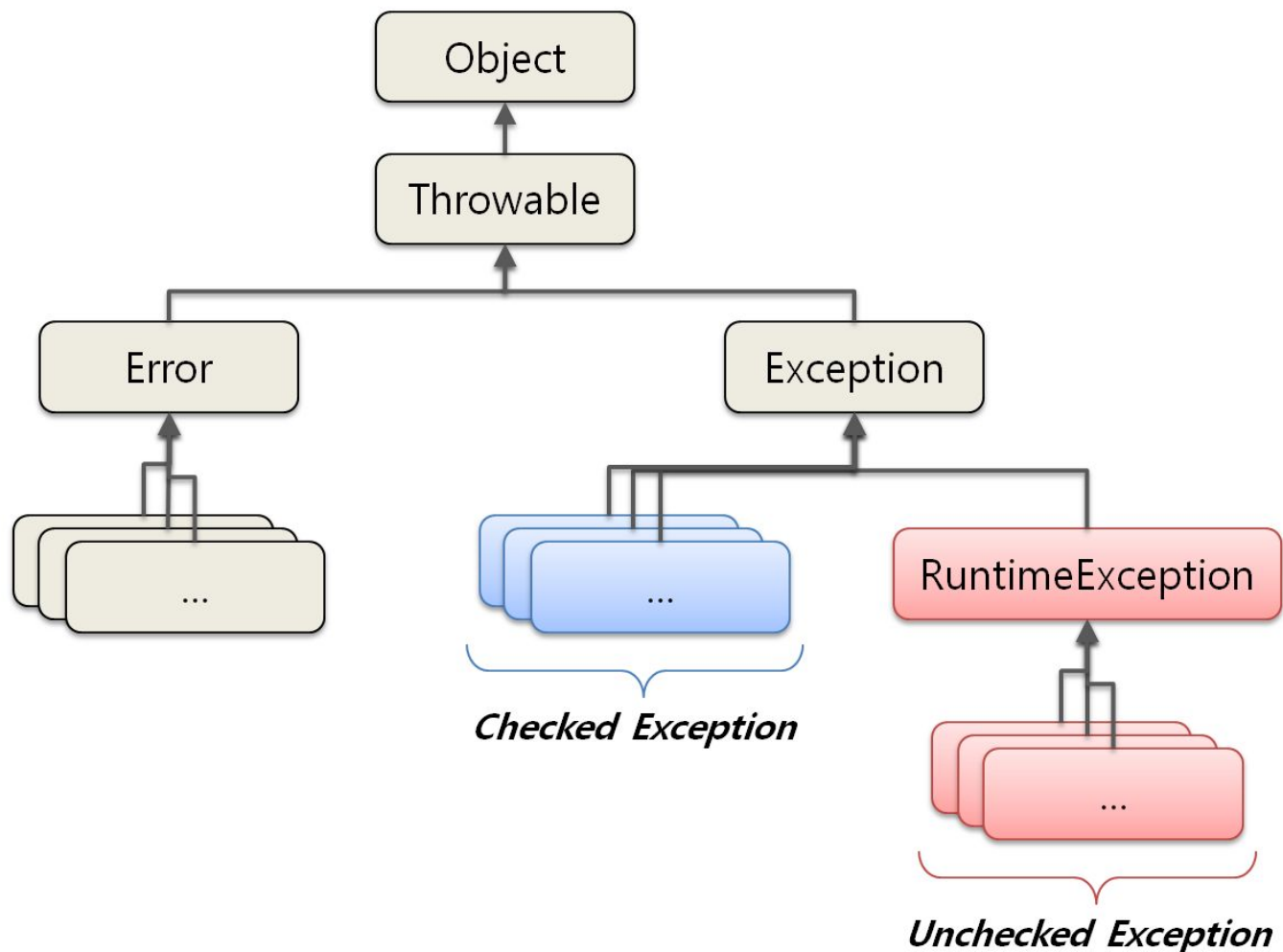


Recap: **FileWriter** Example

- We need to handle exceptions to use the `FileWriter` class.

```
public static void writeLines(String fName, List<String> l) {  
    try {  
        FileWriter fileWriter = new FileWriter(fName);  
        for (String string : l) {  
            fileWriter.write(string);  
        }  
        fileWriter.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

Recap: Exception Object and Hierarchy



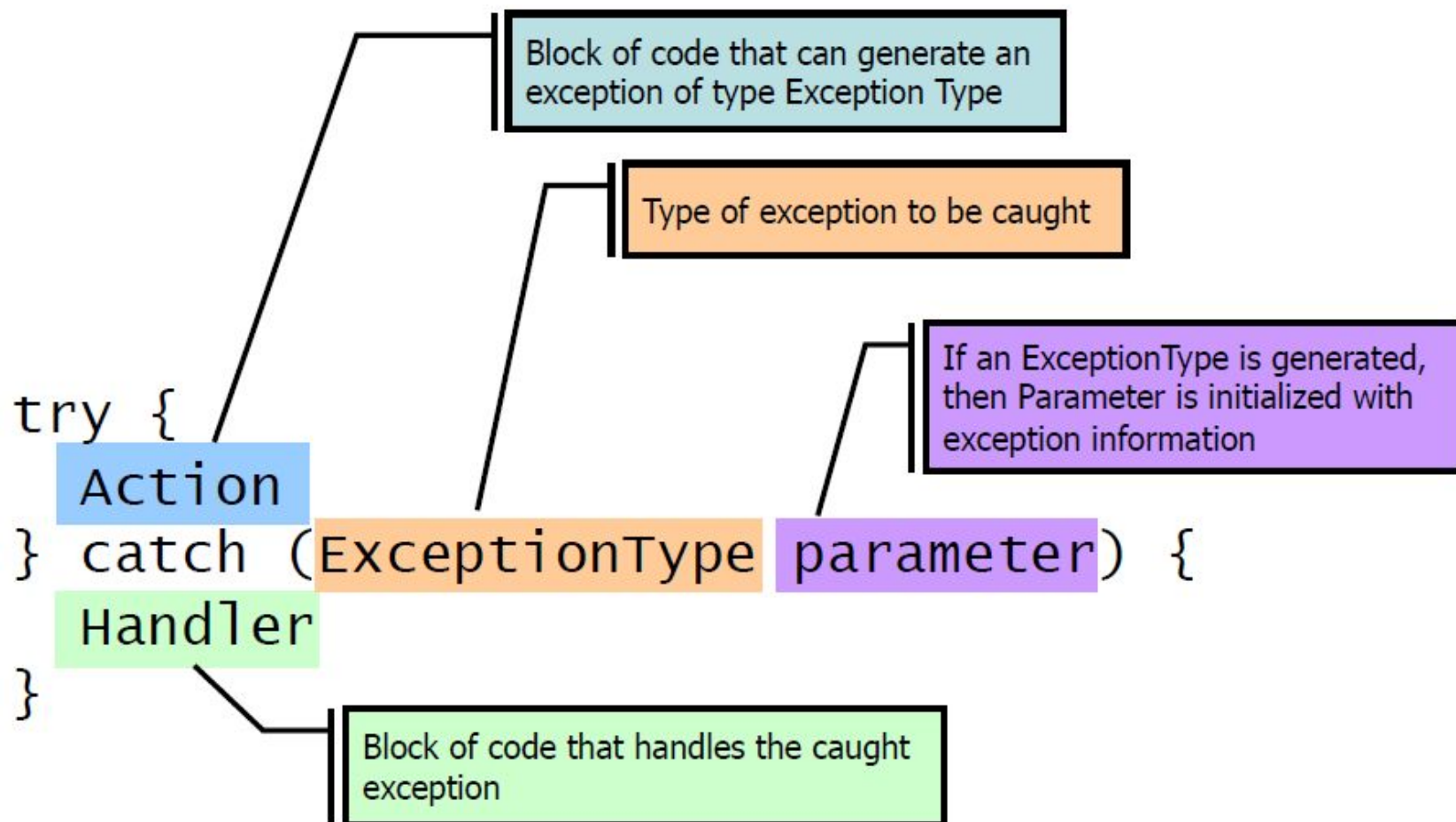
Recap: Example Checked Exceptions

- **IOException**
 - Exceptions related to I/O
 - `FileNotFoundException`, `EOFException`, etc.
- **ClassNotFoundException**
 - Trying to reference the class which corresponding .class file is not found
- **InstantiationException**
 - Trying to initiate abstract class or interface
- **SQLException**
 - Trying to access or query database

Recap: Example RuntimeExceptions

- `NullPointerException`
 - Trying to reference from null value
- `IndexOutOfBoundsException`
 - Accessing index larger than the size of array / string
- `IllegalArgumentException`
 - Abnormal or incorrect arguments for method
- `ArithmeticException`
 - Integer division by zero, etc.
- `ClassCastException`
 - Wrong cast to a class that is not in an inheritance relationship.

Recap: Try-catch Statement



Recap: Try-catch Statement

```
static void test(){  
    Object mObject = null;  
    System.out.println(mObject.getClass());  
}
```

main Method

```
try {  
    test();  
} catch (Exception e){  
    e.printStackTrace();  
}
```

Output

```
java.lang.NullPointerException  
    at main.test(main.java:16)  
    at main.main(main.java:9)
```

Recap: Finally

```
static int thrower(String s) {  
    try {  
        if (s.equals("divide")) {  
            int i = 0;  
            return i / i;           → throws java.lang.ArithmeticException  
        }  
        if (s.equals("null")) {  
            s = null;  
            return s.length();     → throws java.lang.NullPointerException  
        }  
        return 0;  
    } finally {  
        System.out.println( "[thrower(\"" + s + "\") done]");  
    }  
}
```

Codes to execute whether exception happened or not

Recap: Throws, Throw new

```
static void fun() throws IllegalAccessException {  
    System.out.println("Inside fun(). ");  
    throw new IllegalAccessException("demo");  
}
```

main Method

```
try {  
    fun();  
} catch (IllegalAccessException e) {  
    System.out.println("caught in main.");  
}
```

Output

```
Inside fun().  
caught in main.
```

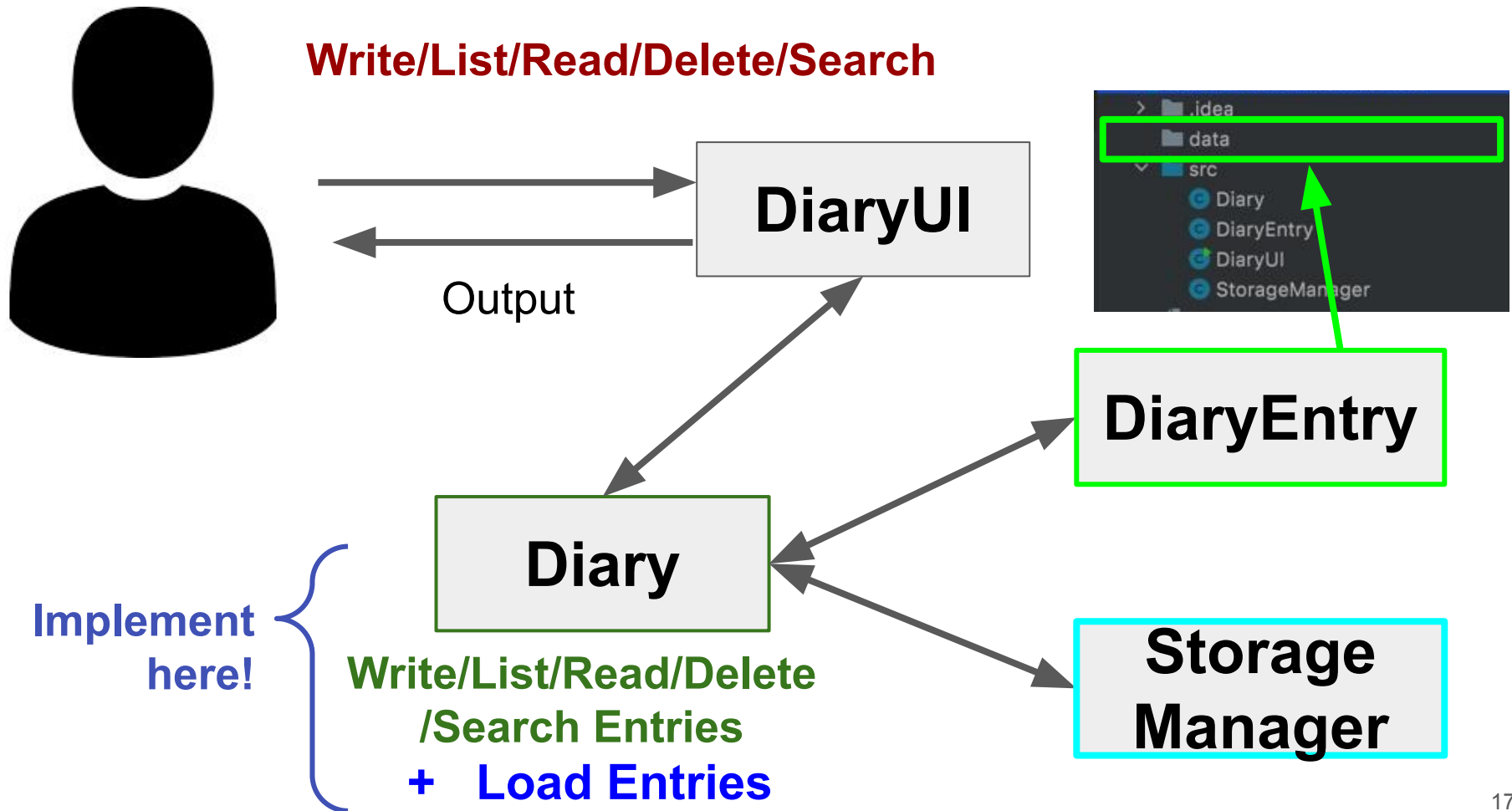
Objectives

- **To properly use JAVA File IO**
- To properly throw custom-made exceptions

Problem Overview

- Problem 1. Simple Diary Application (ver 2)
 - subprob 1 : command *Load Entries* (0:05)
 - subprob 2 : command Create Entry (0:05)
 - subprob 3 : command Delete Entry (0:05)
- Problem 2. Exception Practice
 - subprob 1 : Why NullPointerException? (0:05)
 - subprob 2 : Create custom exception (0:05)
 - subprob 3 : Throw custom exception (0:05)
 - subprob 4 : Graceful termination (0:05)

Problem 1 - Simple Diary Application (2)

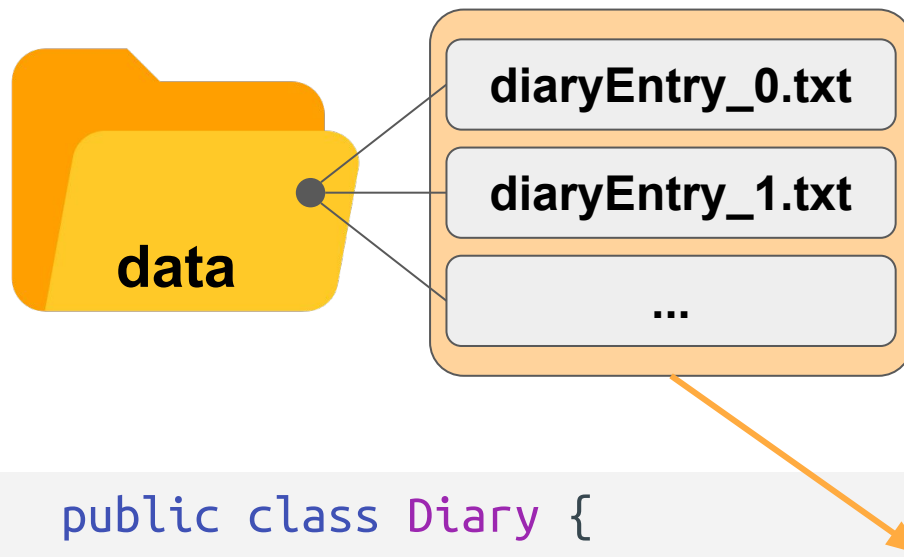


Now, create the *data* directory under lab9

StorageManager Class

- There are methods related to the File I/O jobs.
 - `writeLines`: save string lines into a file
 - `directoryChildrenLines`: read string lines of files in the directory and sort the files by their names.
 - `readLines`
 - `nameSortedDirectoryFiles`
 - `directoryFiles`
 - `deleteFile`: delete a file from the directory

Command 1 - Load Entry



- Load diary entries in *data* directory and store them in the *diaryEntries* List.
- Hint: Use *directoryChildrenLines* in *StorageManager* and *updateIncrementalID* in *DiaryEntry*

```
public class Diary {  
    public List<DiaryEntry> diaryEntries = new LinkedList<>();  
    public Map<Integer, Set<String>> searchMap  
        = new HashMap<>();  
  
    public static String DATA_PATH = "data/";  
  
    public Diary() { loadEntries(); }  
}
```

Command 2 - Create Entry

- Create a diary entry with **title** and **content**.
- Each entry should have its own **unique id** when created and the created time.
- A created diary entry should be stored in a **List**
& in the data directory.
- Assume that title input contains **only** alphanumeric characters and spaces(' ').
- Store the entry in a file using `writeLines` method in `StorageManager` class

```
Command: create  
title: First Entry  
content: Dear Diary, Life is beautiful.  
The entry is saved.
```

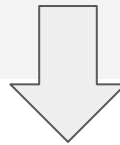
Command 3 - Delete Entry

- **delete <id>** : delete a diary entry with <id>
 - delete it from the List, HashMap and the directory!
- Use *deleteFile* method in StorageManager

Type a command
(...)

Command: **delete 1**

Entry 1 is removed.



Type a command
(...)

Command: **list**

id: 2, created at: 2020/10/21 11:48:30, title: Self Reflection

id: 3, created at: 2020/10/21 11:55:30, title: Third Entry

Objectives

- To properly use JAVA File IO
- **To properly throw custom-made exceptions**

Problem 2 - Exceptions Practice

- Make the DiaryApp (Lab7,8) more **robust**!
- Today's skeleton code is the solution code of the last week. (modified a little)
- We are going to practice how to improve our code design with **Exception**.
- Remove the *data* directory.

Subprob 1 : Why NullPointerException?

- Why NullPointerException?
 - As of now, running the main function of DiaryUI throws a NullPointerException.

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint
    at java.base/java.util.Arrays.sort(Arrays.java:1441)
    at StorageManager.nameSortedDirectoryFiles(StorageManager.java:72)
    at StorageManager.directoryChildrenLines(StorageManager.java:49)
    at Diary.loadEntries(Diary.java:105)
    at Diary.<init>(Diary.java:16)
    at DiaryUI.initializeDiaryUI(DiaryUI.java:8)
    at DiaryUI.main(DiaryUI.java:55)
```

- Find where the source of this problem is in the main function (don't go into deeper call stacks).
- Then make the program end gracefully after printing relevant information ("Diary directory data/ is not found.").

Subprob 1 : Why NullPointerException?

- Expected results
 - Without *data* directory

```
Diary directory data/ is not found.  
  
Process finished with exit code 0
```

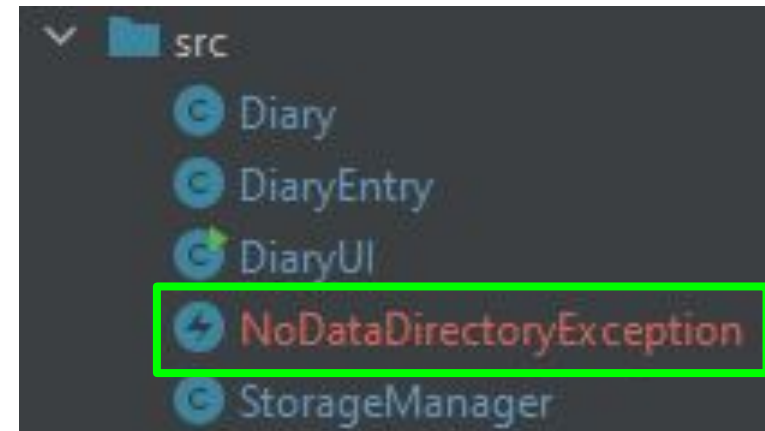
- With *data* directory

```
Type a command  
create: Create a diary entry  
list: List diary entries  
read <id>: Read a diary entry with <id>  
delete <id>: Delete a diary entry with <id>  
search <keyword>: List diary entries whose contents contain  
Command:
```

Subprob 2 : Create custom exception

- Rollback to the original skeleton code, and remove the "data" directory.
- Let's create our own exception class **NoDataDirectoryException** which is raised when there is no "data/" directory.

```
public class NoDataDirectoryException extends Exception{
```



Subprob 2 : Create custom exception

- Find where the `NullPointerException` is raised in `StorageManager` class.
- Throw `NoDataDirectoryException` from that spot, and handle it in the main method of `DiaryUI` class.

Subprob 2 : Create custom exception

- Handle the exception for every method that should handle it.
- Use *throw new exception*

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint
  at java.base/java.util.Arrays.sort(Arrays.java:1441)
  at StorageManager.nameSortedDirectoryFiles(StorageManager.java:66)
  at StorageManager.directoryChildrenLines(StorageManager.java:48)
  at Diary.loadEntries(Diary.java:105)
  at Diary.<init>(Diary.java:16)
  at DiaryUI.initializeDiaryUI(DiaryUI.java:8)
  at DiaryUI.main(DiaryUI.java:57)
```

Subprob 3 : Throw custom exception

- Modify a method ***writeLines*** of **StorageManager** class to throw **NoDataDirectoryException**.
- Handle the exception for every method that should handle it.

Subprob 4 : Graceful termination

- Inform the users of a successful termination of the diary application.
 - Whether the application terminates with an Exception or "exit" command, print the message “Diary application terminated successfully.” and close all resources.

Exception is raised ⇒

```
Diary directory data/ is not found.  
Diary application terminated successfully.
```

Exit command ⇒

```
Type a command  
create: Create a diary entry  
list: List diary entries  
read <id>: Read a diary entry with <id>  
delete <id>: Delete a diary entry with <id>  
search <keyword>: List diary entries whose contents contain <keyword>  
Command: exit  
Diary application terminated successfully.
```

Submission

- Compress your final "src" directory into a **zip** file.
 - After unzipping, the "src" directory must appear.
- Rename your zip file as **20XX-XXXXXX_{name}.zip** - for example, 2021-12345_JeongMinkyung.zip
- Upload it to eTL - Lab 8 assignment.

Thank You!!!

Q&A Time!