You are inside question view of Polymorphism

56m left

Skip to main content

ALL

1

1. Polymorphism

Write three classes:

A Parent class having a filter() method (string type returning null).

Two classes ChildOne and ChildTwo that extend from the class Parent, both overriding the filter() method from the Parent class.

You are given two integers representing the range with start value and end value.

Declare two variables in the Parent class of type int and scope public named startElement and endElement which will represent the start and end element of the range of integers.

The filter() method in the ChildOne class should return a string consisting of all the prime numbers within the given range.

The filter() method in the ChildTwo class should return a string consisting of all the happy numbers within the given range.

Happy numbers are those numbers that return 1 when they are replaced by the sum of the square of the digits repeatedly.

Example:

91 - 9^2 + 1^2

82 - 8^2 + 2^2

68 - 6^2 + 8^2

100 - 1^2 + 0^2 + 0^2

1

The successive addition of squares of the digits of 91 yields 1. Therefore, 91 is a happy number.

Note: A number is considered unhappy when repeatedly the sum of the square of the digits returns 4.

Input Specifications:

The first line is an integer representing the starting number in the range (inclusive).

The second line is an integer representing the ending number in the range (inclusive).

Output Specifications:

The first line should consist of a string with the prime numbers in the range (each number separated by a space).

The second line should consist of a string with the happy numbers in the range (each number separated by a space).

Sample Input:

1

150

Sample Output:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149

1 7 10 13 19 23 28 31 32 44 49 68 70 79 82 86 91 94 97 100 103 109 129 130 133 139

Language: Java 8

Autocomplete Ready

More

1234567891011

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

}

}

Line: 11 Col: 2

Test Results

Custom Input

Run Code

Run Tests

Submit

You are inside question view of ArrayList in Java

* **43m left**

**[Skip to main content](https://www.hackerrank.com/test/eet65dc6mmb/questions/2ja2s12mjik" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/eet65dc6mmb/questions)
* [**1**](https://www.hackerrank.com/test/eet65dc6mmb/questions/2ja2s12mjik)

**1. ArrayList in Java**

A bus has a seating capacity of n. The fare depends on the capacity n and the number of passengers on the bus. For example, if there is less than or equal to 25% of passengers currently on the bus, then they will be charged a traveling fare of n+n\*0.6. If the passenger count is greater than 25% and less than or equal to 50% of the capacity of the bus, then they will be charged a fare of n+n\*0.3. If the number of passengers on the bus is more than 50% of the seating capacity of the bus, then they will be charged a fare of n (If the value on dividing n by 2 or 4 gives a remainder other than 0, then **convert it into its ceiling value**). You will be given the seating capacity, the number of stops during the trip, an arraylist of strings representing the passenger's ID prefixed with a "+" or a "-" sign denoting whether the passenger has got on or out of the bus, respectively, and a query q.

You have to write a program to get the output depending upon the query type. The following are the five type of queries:

* Query type 1 (1): You have to determine the number of passengers who have got on and out of the bus during that trip.
  + Example output: "(number of passengers who have got on the bus) passengers got on the bus and (number of passengers who have got out of the bus) passengers got out of the bus"
* Query type 2 (2): You have to determine the number of passengers who have traveled with a fare of n+n\*0.6, n+n\*0.3, and n.
  + Example output: "(number of passengers who have spent n+n\*0.6 on their fare) passengers traveled with a fare of (n+n\*0.6), (number of passengers who have spent n+n\*0.3 on their fare) passengers traveled with a fare of (n+n\*0.3) and (number of passengers who have spent n as their fare) passengers traveled with a fare of (n)"
* Query type 3 (3, Passenger id): You have to determine how much a particular passenger with the given passenger id has spent during the trip (Note: The same passenger can get on the bus more than once).
  + Example output: "Passenger (Passenger id) spent a total fare of (total fare spent by the passenger during the trip)"
* Query type 4 (4, Passenger id): You have to determine how many times a particular passenger with the given passenger id has got on the bus during the trip.
  + Example output: "Passenger (Passenger id) has got on the bus for (number of times the passenger has got on the bus) times"
* Query type 5 (5, Passenger id): You have to determine whether the given passenger is on the bus or not at the end of the trip
  + Example output: "Passenger (Passenger id) was (inside/not inside depending upon whether the passenger is on the bus or not at the end of the trip) the bus at the end of the trip"

**Note**: In the example outputs, the values enclosed within the parentheses should be replaced with the respective calculated values.

**Input Specifications:**

* The first line of the input consists of an integer n describing the seating capacity of the bus.
* The second line of the input consists of an integer x describing the number of stops during the trip.
* The next x lines of input consist of the strings with the list of passengers who have got on and out of the bus.
* The last line of the input is an integer q describing the query (query types 3, 4, and 5 have a comma and a space separating the values).

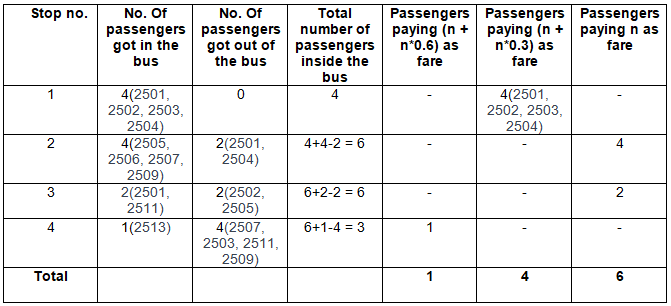
**Output Specifications:**  
    You should print the output depending upon the query given.

**Sample Input 1:**  
10  
4  
+2501 +2502 +2503 +2504  
-2501 -2504 +2505 +2506 +2507 +2509  
+2501 +2511 -2502 -2505  
+2513 -2507 -2503 -2511 -2509  
2

**Sample Output 1:**

1 passengers traveled with a fare of 16.0, 4 passengers traveled with a fare of 13.0 and 6 passengers traveled with a fare of 10.0

**Explanation:**

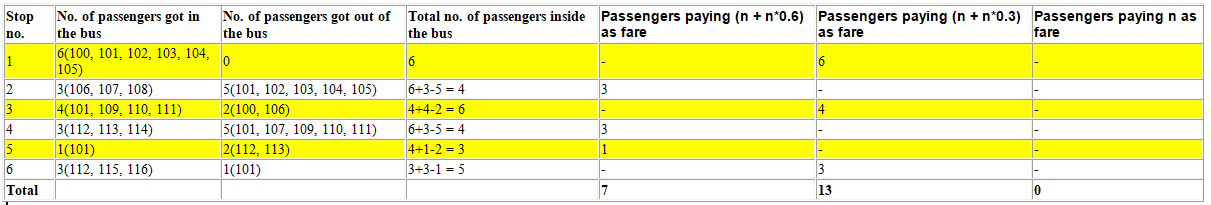


Therefore, in total, 1 passenger spent a fare of 16, 4 spent a fare of 13 and 6 passengers paid a fare of 10.

**Sample Input 2:**  
15  
6  
+100 +101 +102 +103 +104 +105  
+106 +107 +108 -101 -104 -105 -103 -102  
+101 +109 +110 +111 -106 -100  
+112 +113 -101 +114 -109 -107 -110 -111  
+101 -112 -113  
+115 +116 -101 +112  
3, 101

**Sample Output 2:**  
Passenger 101 spent a total fare of 63.0

**Explanation:**



The passenger with id 101 has got on the bus thrice(highlighted in the table). The number of passengers on the bus when the passenger has got on the bus for the first and the second time is greater than 25% of the capacity but less than half of the capacity. Hence, the passenger has spent 19.5 each time. The last time when passenger 101 has got on the bus, there is less than 25% of the capacity of passengers on the bus, so the passenger has spent 24.0. In total, the passenger has spent 63.0 (19.5+19.5+24.0).

**Sample Input 3:**  
15  
6  
+100 +101 +102 +103 +104 +105  
+106 +107 +108 -101 -104 -105 -103 -102  
+101 +109 +110 +111 -106 -100  
+112 +113 -101 +114 -109 -107 -110 -111  
+101 -112 -113  
+115 +116 -101 +112  
5, 101

**Sample Output 3:**  
Passenger 101 was not inside the bus at the end of the trip

**Explanation:**  
    Passenger 101 was not inside the bus at the sixth stop as the last symbol associated with the id is -.

**Note**: You should use an **ArrayList**of the given class type**Passenger**to store the user id and the fare.

 Web IDE

* **Test Results**
* **Help**

Run Tests

Submit

You are inside question view of Exceptions

* **23m left**

**[Skip to main content](https://www.hackerrank.com/test/c6bhpeqe6p9/questions/284nj8i3ln6" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/c6bhpeqe6p9/questions)
* [**1**](https://www.hackerrank.com/test/c6bhpeqe6p9/questions/284nj8i3ln6)

# 1. Exceptions

Shubha wants to send a secret message to her partner Monisha. As part of the design phase, she creates an **Encrypter** class to handle input validation and encryption, and later implements a simple string manipulation method as a placeholder for real encryption.

Define a user-defined exception to the **InvalidMessageException** class which extends the **Exception** class.

Implement an **Encrypter** class which implements the **String encryptMessage(String name) static** method.

The method must perform the following tasks:

* Should validate the message by calling the **validate(message)** method provided in the **Validator** class.
* If message validation is successful, the message string should be reversed, and the uppercase letters must be converted to lowercase letters. This new string should then be returned as the encrypted message.
* If message validation fails, the method should display an exception of the **InvalidMessageException** class with the message **InvalidMessageException: Try again with valid message**. For custom messages, overload the **toString** method in **InvalidExceptionMessage**with your custom message.

The locked stub code in the editor provides complete implementation of the **Validator** class, and validates the implementation of the **Encrypter** class.

**Constraints:**

* The message should not contain more than **100** characters.

**Input Format for Custom Testing:**

Encrypted name

**Sample Case 0**

**Sample Input:**

    Kate Winslet

**Sample Output:**

    telsniw etak

**Explanation:**

    The name validation is successful, therefore, the string is reversed and converted to lowercase.

**Sample Case 1**

**Sample Input:**

    Kate Wins?let

**Sample Output:**

    InvalidMessageException: Try again with valid message

**Explanation:**

    Invalid characters were found in the name, therefore, the exception is displayed with the message **Try again with valid  
    message**.

Language: Java 8Autocomplete Ready

More

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

**import** java.util.Scanner;

**class** Validator {

**public** **static** **boolean** validate(String message) {

**return** message.matches("[A-Za-z0-9 ]+");

    }

}

**public** **class** Solution {

**private** **static** **final** Scanner INPUT\_READER = **new** Scanner(System.in);

**public** **static** **void** main(String[] args) {

        String message = INPUT\_READER.nextLine();

**try** {

            String encrypted\_message = Encrypter.encryptMessage(message);

**if**(! encrypted\_message.startsWith("InvalidMessageException"))

                System.out.println(encrypted\_message);

        } **catch** (Exception e) {

            System.out.println(e);

        }

    }

}



Line: 6 Col: 43

Test Results

Custom Input

Run Code

Run Tests

Submit

You are inside question view of HashMap Library

* **54m left**

**[Skip to main content](https://www.hackerrank.com/test/1cml6ap4s01/questions/3d6di3qierl" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/1cml6ap4s01/questions)
* [**1**](https://www.hackerrank.com/test/1cml6ap4s01/questions/3d6di3qierl)

# 1. HashMap Library

**COLLECTIONS IN JAVA – HASHMAP – LIBRARY**

In a library, there is a set of books from which some are borrowed by the users. You have to perform certain operations with respect to the query using **hashmap** in Java.

You will be given three datasets. The first one (**booksInLibrary**) contains **book id**, **book name**, and **book author**. The second (**borrowedUsers**) contains **book id** and **user id**. The third (**query**) contains the **query** to perform. These datasets are of string type and are passed in the **getQuery()** function. Split the string and get the respective parameters.

You have to implement the following three functions in class library:

* createLibraryMap(String booksInLibrary)
* createUserMap(String borrowedUsers)
* getQuery(String booksInLibrary,String borrowedUsers,String query)

**Query Type 1:**

            Check the availability of a book using its book id. The input should be the **book id**. If the book is present, the result should be "It is available\nAuthor is <author name>\n". If the book is not present and borrowed by somebody else, the result should be "No Stock\nIt is owned by <user id>\n".

**Query Type 2:**

            Return the books borrowed by a specific user by using their user id. The input should be **user id**, and the output should be "<book id> <bookname>\n.... and so on" .

**Query Type 3:**

            There is more than one book for a given book name. Return the number of books present and borrowed by a user for a specific book name. The input should be the **book name**. The result should be "<no. of books borrowed> out\n<no. of books present> in\n".

**Query Type 4:**

            Return the book names of a specific author. The input should be the **book author**. The output should be "<book name>\n.....and so on".

**Query Type 5:**

            Return the book names based on the given keyword. The input should be the search keyword. The result should be "<book id> <book name>\n".

**Sample Input**

**booksInLibrary** = 125,C programming,Brian W. Kernighan|546,Java programming,James Gosling|897,Data Structure,Adam Drozdek|265,Data Structure,Adam Drozdek|234,The Java Language Specification,James Gosling

125 - book id , C programming - book name , Brian W. Kernighan - author name

**borrowedUsers** = 125,101|897,104

125 - book id , 101 - user id.

***Note*:**In **booksInLibrary,**books are separated by "|", book details (book id, book name, author name) are separated by ",". In **borrowedUsers,**the users are separated by "|", user details (book id, user id) are separated by ",". In a **query** string, the first character refers to the query type, and the second character refers to the conditional string depending on the query type.

**For query type 1**: **query =**1,546 (546 - book id)

**Sample output:**

No Stock

It is owned by 101

**For query type 2**: **query =**2,101 (101 - user id)

**Sample output:**

546 Java programming

125 C programming

**For query type 3**: **query =**3,Data Structure (Data Structure - book name)

**Sample output:**

1 out

1 in

**For query type 4:**query**=**4,James Gosling (James Gosling - author name)

**Sample output**

Java programming

The Java Language Specification

**For query type 5: query =**5,java (java - search string)

**Sample output:**

546 Java programming

234 The Java Language Specification

In **createLibraryMap(String booksInLibrary)**, you have to return the hashmap of the book details. The key should be **book id**, and the value should be an object which holds the book name and author.

In **createUserMap(String borrowedUsers)**, you have to return the hashmap of the user details. The key should be **book id**, and the value should be the **user id**.

In **getQuery(String booksInLibrary,String borrowedUsers,String query)**, you have to return the result based on the query.

 Web IDE

* **Test Results**
* **Help**

Run Tests

Submit

You are inside question view of HashSet in Java

* **58m left**

**[Skip to main content](https://www.hackerrank.com/test/63tptctokop/questions/2ja30kbag67" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/63tptctokop/questions)
* [**1**](https://www.hackerrank.com/test/63tptctokop/questions/2ja30kbag67)

# 1. HashSet in Java

A football team has played n matches in a tournament. They have played each match with different sets of players (some players may have continuously played in some of the matches). You have to write a program to find the players who have played all the matches and the players who have played in a particular match and not in the other mentioned match.

**Input Specifications:**

* The first line consists of an integer n denoting the number of matches.
* The second line consists of a single string containing the squad of players in each match, separated by a #. The players in a particular match are separated by a single space (the players for a single match may contain duplicates as well, you need to remove the duplicates).
* The last two lines consist of two integers x and y, which denotes the match numbers depending on which you have to find the players who have played in match y, but not in x.

**Output Specifications:**

Output a string with the players who have played all the matches with the names separated by a single space, followed by a comma and space, and the players who have played match number y and have not played match number x with the names separated by a single space.

**Sample Input:**  
4  
Ronaldo Eder Andre Bruma Goncalo Jota Daniel Patrício Ricardo Domingos Semedo Bernardo Pereira Bruno Pizzi Paciencia Jota Daniel Patrício Eder Andre Bruma Goncalo Jota Daniel Patrício Ricardo Domingos Semedo Bernardo Pereira Bruno#Ronaldo Eder Andre Bruma Goncalo Joao Dyego Jose Nelson Pepe Mario Pereira Rafa Renato Ferro Bruma Goncalo Joao Dyego Pereira Rafa Renato Ferro Bruma Goncalo Joao Dyego#Ronaldo Eder Andre Bruma Goncalo Jota Daniel Patrício Nelson Pepe Mario Pereira Rafa Renato Ferro Daniel Patrício Daniel Patrício Nelson Pepe Mario Pereira Nelson Pepe Mario Pereira Rafa Renato Ferro Daniel Patrício#Ronaldo Eder Andre Bruma Goncalo Joao Dyego Jose Ricardo Domingos Semedo Bernardo Pereira Bruno Pizzi Andre Bruma Goncalo Joao Dyego Jose Ricardo Domingos Semedo Bernardo  
2  
4

**Sample Output:**  
Pereira Ronaldo Goncalo Eder Andre Bruma, Ricardo Pizzi Semedo Bruno Domingos Bernardo

**Explanation:**  
Players Pereira, Ronaldo, Goncalo, Eder, Andre, and Bruma have played in all 4 matches. Players Ricardo, Pizzi, Semedo, Bruno, Domingos and Bernardo have played in the 4th match but not in the 2nd.

**Note:**

* You have to use HashSet for this program.
* Since you are using HashSet, the order of the names of the players in the output can be in any order.

 Web IDE

* **Test Results**
* **Help**

Run Tests

Submit

You are inside question view of TreeMap Cricket

* **59m left**

**[Skip to main content](https://www.hackerrank.com/test/8lfj821np2/questions/97e9erie7rg" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/8lfj821np2/questions)
* [**1**](https://www.hackerrank.com/test/8lfj821np2/questions/97e9erie7rg)

# 1. TreeMap Cricket

**COLLECTIONS IN JAVA – TREEMAP – CRICKET**

There is a set of cricket player details for matches. You have to perform certain operations with respect to the query using TreeMap in Java.

You will be given two datasets. For the query types 1 and 2, the first parameter (**cricketDataset**) contains **position**of the player in the match**, name of the player and the score of the player in a match**. For the query type 3, the first parameter (**cricketDataset**) contains the **number of matches**. The second parameter (**query**) contains the **query** to perform. These datasets are of string type and are passed using the **getQuery()** function. Split the string and to get the respective parameters.

You have to implement the following three functions in class library:

* createPlayerPositionMap(String cricketDataset)
* createPlayerScoreMap(String cricketDataset)
* createMatchesMap(String cricketDataset)
* getQuery(String cricketDataset,String query)

**Query Type 1:**

            Return the list of players with position and their names from starting position to end position. Input should be start position and end position. The result should be "<player position> <player name>\n...and so on".

**Query Type 2:**

            Return the list of players with position and their names more than specified threshold score. Input should be threshold score. Your result should be "<player position> <player name>\n...and so on".

**Query Type 3:**

            Return the efficient opener of all the matches depends upon the average score score of a player by calculate the total score of all matches divided by number of matches played. Your result should be "The Efficient Opener is <player name>".

***Note:*** The opener is the player who played first position in a match.

**Sample Input:**

**cricketDataset for query type 1 and 2   =**  3,Dhoni,120|1,Virat,103|5,Jadeja,40|2,Rohit,70|4,Pandya,30

3 - position of the player, Dhoni - name of the player, 120 - score of the player

***Note*:**In **cricketDataset,**players are seperated by "|", player details(player's position, name and score) are seperated by ",". In a **query** string, the first character refers the query type, the second character refers the conditional string depends on the query type. For query

**Type 1**: **query =**1 3 5 (3 - start position, 5 - end position)

**Sample output:**

3 Dhoni

4 Pandya

5 Jadeja

**Type 2**: **query =**2 50 (50 - threshold score)

**Sample output :**

1 Virat

2 Rohit

3 Dhoni

**cricketDataset for query type 3 =**3,Rohit,100|2,Virat,56|1,Dhoni,150\n3,Rahul,90|2,Virat,100|1,Rohit,99\n1,Rahul,50|2,Virat,64|3,Rohit,78\n1,Dhoni,95|2,Virat,30|3,Rohit,45\n3,Jadeja,43|2,Virat,54|1,Rohit,40|4,Dhoni,59

**Calculation:**

Dhoni - 150+95/2 - 122.5

Rohit - 95+78+40/3 - 71

Rahul - 50/1 - 50

**Type 3**: **query =**3 (only type of the query will be given)

**Sample output:**

The Efficient Opener is Dhoni

Use **createPlayerPositionMap(String cricketDataset)** to return the TreeMap of query type 1. The key should be the position of the player, and the value should be the name of the player.

Use **createPlayerScoreMap(String cricketDataset)** to return the TreeMap of query type 2. The key should be the name of the player, and the value should be the score of the player.

Use **createMatchesMap(String cricketDataset)** to return the TreeMap of query type 3.

Use **getQuery(String cricketDataset,String query)**to return the result string with respect to the query type.

 Web IDE

* **Test Results**
* **Help**

Run Tests

Submit

You are inside question view of Working with Multithreading

* **55m left**

**[Skip to main content](https://www.hackerrank.com/test/1b6tg03f7dn/questions/42fj3aqjage" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/1b6tg03f7dn/questions)
* [**1**](https://www.hackerrank.com/test/1b6tg03f7dn/questions/42fj3aqjage)

# 1. Working with Multithreading

You will be given an array of size 300. The name of the array is threadArray, and the index of the array should be referenced by using a variable name i.

(Note: The variables threadArray, and i (variable name to access the index of the array) are declared in the predefined class solution).

You have to insert elements into the array using three threads by creating three classes, namely Task1, Task2, and Task3, respectively. The values to be inserted into the array are 0,1,2,3,.....299.

 I=80 j=130 k=90

0-79

80-209

210-299

You should override the run method in the threads. Three integers i, j, and k will be given, representing the number of elements each thread should append inside the given array.  
Thread one should append 0 to i-1 inside the array, thread two should append i to i+j-1 inside the array, and the third thread should append i+j to 299 inside the array.  
Threads one and two must run simultaneously, and the values of the threads one and two must be inserted inside the indices of the array from 0 to i+j-1 randomly. The third thread should start only after the first two threads have been executed completely.

**Input Specifications:**  
The first line is an integer representing the number of elements to be inserted inside the array by thread one.  
The second line is an integer representing the number of elements to be inserted inside the array by thread two.  
The third line is an integer representing the number of elements to be inserted inside the array by thread three.

**Output Specifications:**  
The output should be true if the processes are executed in the correct order. Otherwise, it should be false.

**Sample Input:**  
80  
130  
90

**Sample Output:**  
true (if both the threads one and two have run simultaneously and the third thread has run after them), or false (if threads one and two have not run simultaneously and thread three has started before threads one and two have got completed).

Language

Java 8



Autocomplete Ready

More

22

23

24

25

26

27

28

29

14

15

16

17

18

19

20

21

6

7

8

9

10

11

12

13

1

2

3

4

5

        String oneAndTwo = "";

        String sizeOfTask1 = "";

**for**(**int** i=0;i<first;i++)

        {

            oneAndTwo += threadArray[i]+" ";

        }

**for**(**int** i=0;i<containsSecondThread;i++)

        {

        Thread task3Thread = **new** Thread(task3);

        task1.start();

        task2Thread.start();

        task1.join();

        task2Thread.join();

        task3Thread.start();

**int** first = Task1.a+Task2.a;

**int** containsSecondThread = Task1.a;

**public** **static** **final** **int**[] threadArray = **new** **int**[300];

**public** **static** **volatile** String i = 0+"";

**public** **boolean** test() **throws** InterruptedException

    {

        Task1 task1 = **new** Task1();

        Task2 task2 = **new** Task2();

        Task3 task3 = **new** Task3();

        Thread task2Thread = **new** Thread(task2);

**import** java.util.Scanner;

*//Write your code here*

**public** **class** Solution {



Line: 64 Col: 2

Test Results

Custom Input

Run Code

Run Tests

Submit

You are inside question view of File Operations

* **56m left**

**[Skip to main content](https://www.hackerrank.com/test/78g8q6jhan5/questions/b2pjft6rchk" \l "main-content)**

* [**ALL**](https://www.hackerrank.com/test/78g8q6jhan5/questions)
* [**1**](https://www.hackerrank.com/test/78g8q6jhan5/questions/b2pjft6rchk)

# 1. File Operations

Nandhini wants to store a decrypted message, and read an encrypted message sent by her friend. As part of the design phase, Nandhini creates an **EncryptDecryptFile**class to store the decrypted message in the **DecryptionFile.txt**fileand to read the encrypted message in the **EncryptionFile.txt**file.

Implement the **EncryptDecryptFile**class which must implement the **public void writeDecryptionFile(String message)** and **public String readEncryptionFile()**methods.

**void writeDecryptionFile(String message):**  Create and write the **DecryptionFile.txt**file in the specified file path (must contain the decrypted message passed).

**String readEncryptionFile():**Read the encrypted message from the **EncryptionFile.txt**file in the specified file path (must return the encrypted message).

***Note:***Accessing the ***Solution.filepath*** variable to know the file path.

The locked code stub validates the implementation of the **EncryptDecryptFile**class.

**Input Format:**

The first line contains a decrypted message, which must be stored in the **DecryptionFile.txt**file***.***

**Sample Input:**

Hello World!!!

**Sample Output:**

Hello World!!!

**Output Specification:**

The first line contains a decrypted message, which is stored in the **DecryptionFile.txt**file***.***

Language: Java 8Autocomplete Ready

More

1

4

5

6

7

8

**import** java.io.\*;

*// Write your code below.*

*//*

**public** **class** Solution {



Line: 8 Col: 24

Test Results

Custom Input

Run Code

Run Tests

Submit