QUESTION 1:

Library Book Borrowing Tracker Using HashMap

You are building a system to track books borrowed from a library. The system should store the titles of books along with the number of times they have been borrowed. Users should be able to add, update, or remove book records, check if a book exists, and retrieve specific borrow counts. Additionally, the system should calculate the total number of books borrowed from the library.

If any operation attempts to access a book that does not exist in the system, a custom exception, BookNotFoundException, should be thrown and handled appropriately.

Operations to be Performed

1. **Add Book Records**:
   Input the number of books and their respective titles and borrow counts.
2. **Retrieve Borrow Count**:
   Look up the total number of times a specific book has been borrowed by its title.
3. **Remove Book Records**:
   Remove a book's record from the system.
4. **Check if a Book Exists**:
   Verify if a specific book is in the system based on its title.
5. **Calculate Total Borrow Count**:
   Display the total number of times books have been borrowed.

Input Format

Adding Books:

- The first line contains an integer n, representing the number of books to add.
- For each book, input:
    - The first line contains the book’s title (String).
    - The second line contains the borrow count (Integer).

Perform Operations:

- A line containing the title of a book to retrieve its borrow count.
- A line containing the title of a book to remove from the system.

- A line containing the title of a book to check if it exists in the system.

Output Format

Retrieving Borrow Count:

- If found, print: [Book title] has been borrowed [count] times.
- If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found.

Removing a Book:

- If found and removed, print: Book [Book title] has been removed from the library.
- If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found for removal.

Checking if a Book Exists:

- If the book exists, print: Book [Book title] exists in the library.
- If the book doesn't exist, throw and catch the BookNotFoundException and print: Book [Book title] does not exist in the library.

Calculating Total Borrow Count:

- Print the total borrow count of all books: Total Books Borrowed: [total count]

Sample Input 1

3

The Alchemist

150

To Kill a Mockingbird

200

1984

100

The Alchemist

To Kill a Mockingbird

1984

Sample Output 1

The Alchemist has been borrowed 150 times.

Book To Kill a Mockingbird has been removed from the library.

Book 1984 exists in the library.

Total Books Borrowed: 250

Sample Input 2

2

The Great Gatsby

250

Pride and Prejudice

300

Moby Dick

War and Peace

Anna Karenina

Sample Output 2

mathematica

Copy code

Book Moby Dick not found.

Book War and Peace not found for removal.

Book Anna Karenina does not exist in the library.

Total Books Borrowed: 550

QUESTION 2:

Bachelor Party Gift Management System Using HashSet

You are building a system to manage the gift inventory for a bachelor party. Each gift has a unique ID, a name, and a status to indicate whether it has been delivered or not. The system needs to:

1. Add new gifts to the inventory.
2. Mark gifts as delivered by their ID.
3. Remove gifts from the inventory by their ID.
4. Track how many gifts are still pending delivery after each operation.

Task:

Implement a BachelorPartyGiftSystem class using a HashSet to manage the list of gifts. Each gift should be represented by a Gift class with the following attributes:

- id (int): A unique identifier for the gift, starting from 501 and increasing sequentially for each new gift added.
- name (String): The name of the gift.
- delivered (boolean): Status indicating whether the gift has been delivered or not. (default = false)

The BachelorPartyGiftSystem class should include the following methods:

Functionalities

Add Gift

- Implement the method public void addGift(String name) to add a new gift to the inventory list.
- The gift's ID should be assigned automatically, starting from 501 and incrementing for each new gift.
- This method should take the gift name as input and create a new Gift object with a unique ID and the given name, then add it to the list of gifts.

Deliver Gift

- Implement the method public void deliverGift(int id) to mark a gift as delivered by its ID.
- If the gift is found, mark it as delivered and print a message indicating that the gift with the given ID has been delivered.
- If the gift is not found, print a message indicating that the gift with the given ID is not found.

Remove Gift

- Implement the method public void removeGift(int id) to remove a gift from the inventory by its ID.
- Print whether the gift was successfully removed or if it was not found.

Track Pending Gifts

- Implement the method public int countPendingGifts() to count and return the number of gifts that are still pending delivery.
- This method will be used to display how many gifts are still pending after performing the deliver and remove operations.

Exception Handling

- In case a gift is not found during the Deliver or Remove operations, a custom exception GiftNotFoundException should be thrown.
- This exception should be caught in the methods where the operation is performed and should print a message indicating that the gift with the given ID could not be found.

Input Format

Adding Gifts:

- The first line contains an integer n, representing the number of gifts to add to the inventory.
- For each gift, input:
    - The name of the gift (String).

Perform Operations:

- A line containing the ID of the gift to mark as delivered.
- A line containing the ID of the gift to remove from the inventory.

Output Format

Current Inventory:

- Print "Gifts in the Inventory:"
- For each gift in the inventory, print the gift details in the format: Gift{id=<id>, name='<name>', delivered=<delivered>}

Gift Delivery Status:

- If the gift is delivered, print: Gift with ID <id> has been delivered.
- If the gift is not found, throw and catch the GiftNotFoundException and print: Gift with ID <id> not found.

Gift Removal Status:

- If the gift is removed, print: Gift with ID <id> removed successfully.
- If the gift is not found, throw and catch the GiftNotFoundException and print: Gift with ID <id> not found.

Updated Inventory:

- Print "Updated Inventory:"
- For each gift in the updated inventory, print the gift details in the format: Gift{id=<id>, name='<name>', delivered=<delivered>}

Total Pending Gifts:

- Print "Total pending gifts: <number>", where <number> is the total count of gifts that are still pending delivery.

Sample Input 1

3

Wine Bottle

Groom’s Hat

Custom Mug

502

503

Sample Output 1

Gifts in the Inventory:

Gift{id=501, name='Wine Bottle', delivered=false}

Gift{id=502, name='Groom’s Hat', delivered=false}

Gift{id=503, name='Custom Mug', delivered=false}

Gift with ID 502 has been delivered.

Gift with ID 503 removed successfully.

Updated Inventory:

Gift{id=501, name='Wine Bottle', delivered=false}

Gift{id=502, name='Groom’s Hat', delivered=true}

Total pending gifts: 1

Sample Input 2

4

Party Popper

Dance Shoes

Sunglasses

Cufflinks

510

508

Sample Output 2

Gifts in the Inventory:

Gift{id=501, name='Party Popper', delivered=false}

Gift{id=502, name='Dance Shoes', delivered=false}

Gift{id=503, name='Sunglasses', delivered=false}

Gift{id=504, name='Cufflinks', delivered=false}

Gift with ID 510 not found.

Gift with ID 508 not found.

Updated Inventory:

Gift{id=501, name='Party Popper', delivered=false}

Gift{id=502, name='Dance Shoes', delivered=false}

Gift{id=503, name='Sunglasses', delivered=false}

Gift{id=504, name='Cufflinks', delivered=false}

Total pending gifts: 4