JAVA LABORATORY WEEK-7

PROGRAM ON PACKAGES AND EXCEPTIONS

1. Build a simple library management system in Java, organize your code into three packages: library.books, library.members, and library.transactions. In the library.books package, create a Book class with attributes like title, author, and ISBN, and provide methods to access these attributes. In the library.members package, define a Member class with attributes such as name and member ID, along with methods to retrieve these details. In the library.transactions package, implement a LibraryTransaction class with instance methods to manage borrowing and returning books, allowing you to maintain and manipulate transaction records. Finally, in your main application, create instances of Book, Member, and LibraryTransaction, and use these instances to perform transactions and print the details to demonstrate the system's functionality.

Sample I/O:

Enter details for book 1:

Title: To Kill a Mockingbird

Author: Harper Lee

ISBN: 9780060935467

Enter details for book 2:

Title: 1984

Author: George Orwell

ISBN: 9780451524935

Enter details for member 1:

Name: Alice Smith

Member ID: M001

Enter details for member 2:

Name: Bob Johnson

Member ID: M002

Choose an action:

1. Borrow a book

2. Return a book

3. Exit

1

Select a member by ID: M001

Select a book by ISBN: 9780060935467

Alice Smith borrowed the book: To Kill a Mockingbird

Choose an action:

1. Borrow a book

2. Return a book

3. Exit

2

Select a member by ID: M002

Select a book by ISBN: 9780451524935

Bob Johnson returned the book: 1984

PROGRAM:

Book.java

```
package Library;

public class Book {
    private String title,author;
    private String isbn;
    private boolean isBorrowed;

    public Book(String t, String a, String i) {
    title = t;
        author = a;
        isbn = i;
        isBorrowed = false;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
```

```
        return author;
    }

    public String getIsbn() {
        return isbn;
    }

    public boolean isBorrowed() {
        return isBorrowed;
    }

    public void borrowBook() {
        this.isBorrowed = true;
    }

    public void returnBook() {
        this.isBorrowed = false;
    }
}
```

Member.java

```
package Library;

public class Member {
    private String name,memberId;

    public Member(String n, String m) {
        name = n;
        memberId = m;
    }

    public String getName() {
        return name;
    }

    public String getMemberId() {
        return memberId;
    }
}
```

Transaction.java

```
package Library;

import Library.*;
```

```
public class Transaction {
    private Book[] borrowedBooks;
    private int borrowedCount;

    public Transaction(int capacity) {
        borrowedBooks = new Book[capacity];
        borrowedCount = 0;
    }

    public void borrowBook(Member member, Book book) {
        if (!book.isBorrowed()) {
            if (borrowedCount < borrowedBooks.length) {
                book.borrowBook();
                borrowedBooks[borrowedCount++] = book;
                System.out.println(member.getName() + " borrowed the book: " + book.getTitle());
            } else {
                System.out.println("Cannot borrow more books, transaction limit reached.");
            }
        } else {
            System.out.println("Sorry, the book is already borrowed.");
        }
    }

    public void returnBook(Member member, Book book) {
        boolean found = false;
        for (int i = 0; i < borrowedCount; i++) {
            if (borrowedBooks[i].getIsbn().equals(book.getIsbn())) {
                book.returnBook();
                System.out.println(member.getName() + " returned the book: " + book.getTitle());

                // Shift remaining books to the left to fill the gap
                for (int j = i; j < borrowedCount - 1; j++) {
                    borrowedBooks[j] = borrowedBooks[j + 1];
                }
                borrowedBooks[--borrowedCount] = null;
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("The book was not borrowed by this member.");
        }
    }
```

```
    public void printBorrowedBooks() {
        System.out.println("Currently borrowed books:");
        for (int i = 0; i < borrowedCount; i++) {
            System.out.println(borrowedBooks[i].getTitle());
        }
    }
}
```

class LibraryManagementSystem.java

```
import Library.*;

import java.util.Scanner;

public class LibraryManagementSystem {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Book[] books = new Book[2];
        Member[] members = new Member[2];
        Transaction transaction = new Transaction(2);

        // Input for books
        System.out.println("Enter details for book 1:");
        books[0] = createBook(scanner);
        System.out.println("Enter details for book 2:");
        books[1] = createBook(scanner);

        // Input for members
        System.out.println("Enter details for member 1:");
        members[0] = createMember(scanner);
        System.out.println("Enter details for member 2:");
        members[1] = createMember(scanner);

        // Menu-driven interaction
        boolean exit = false;
        while (!exit) {
        System.out.println("-------------------------");
            System.out.println("Choose an action:");
            System.out.println("1. Borrow a book");
            System.out.println("2. Return a book");
            System.out.println("3. Exit");
        System.out.println("-------------------------");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline
```

```
        switch (choice) {
            case 1:
                System.out.print("Select a member by ID: ");
                Member borrowingMember = selectMember(scanner.nextLine(), members);
                if (borrowingMember != null) {
                    System.out.print("Select a book by ISBN: ");
                    Book borrowingBook = selectBook(scanner.nextLine(), books);
                    if (borrowingBook != null) {
                        transaction.borrowBook(borrowingMember, borrowingBook);
                    }
                }
                break;
            case 2:
                System.out.print("Select a member by ID: ");
                Member returningMember = selectMember(scanner.nextLine(), members);
                if (returningMember != null) {
                    System.out.print("Select a book by ISBN: ");
                    Book returningBook = selectBook(scanner.nextLine(), books);
                    if (returningBook != null) {
                        transaction.returnBook(returningMember, returningBook);
                    }
                }
                break;
            case 3:
                exit = true;
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}

private static Book createBook(Scanner scanner) {
    System.out.print("Title: ");
    String title = scanner.nextLine();
    System.out.print("Author: ");
    String author = scanner.nextLine();
    System.out.print("ISBN: ");
    String isbn = scanner.nextLine();
    return new Book(title, author, isbn);
}

private static Member createMember(Scanner scanner) {
    System.out.print("Name: ");
    String name = scanner.nextLine();
    System.out.print("Member ID: ");
```

```
        String memberId = scanner.nextLine();
        return new Member(name, memberId);
    }

    private static Member selectMember(String memberId, Member[] members) {
        for (Member member : members) {
            if (member.getMemberId().equals(memberId)) {
                return member;
            }
        }
        System.out.println("Member not found.");
        return null;
    }

    private static Book selectBook(String isbn, Book[] books) {
        for (Book book : books) {
            if (book.getIsbn().equals(isbn)) {
                return book;
            }
        }
        System.out.println("Book not found.");
        return null;
    }
}
```

OUTPUT:

```
Enter details for book 1:
Title: Harry Potter
Author: J.K.Rowling
ISBN: 123456
Enter details for book 2:
Title: War and Peace
Author: Leo Tolstoy
ISBN: 456789
Enter details for member 1:
Name: Dinesh
Member ID: 101
Enter details for member 2:
Name: Ram
Member ID: 102
------------------------
Choose an action:
1. Borrow a book
2. Return a book
3. Exit
```

```
------------------------
1
Select a member by ID: 103
Member not found.
------------------------
Choose an action:
1. Borrow a book
2. Return a book
3. Exit
------------------------
1
Select a member by ID: 101
Select a book by ISBN: 123456
Dinesh borrowed the book: Harry Potter
------------------------
Choose an action:
1. Borrow a book
2. Return a book
3. Exit
------------------------
2
Select a member by ID: 101
Select a book by ISBN: 456789
The book was not borrowed by this member.
------------------------
Choose an action:
1. Borrow a book
2. Return a book
3. Exit
------------------------
2
Select a member by ID: 101
Select a book by ISBN: 123456
Dinesh returned the book: Harry Potter
------------------------
Choose an action:
1. Borrow a book
2. Return a book
3. Exit
------------------------
3
```

2. Develop a simple ATM system where users must enter a 4-digit PIN to access their accounts. Implement a system that validates the entered PIN to ensure it is exactly 4 digits long and contains only numeric characters, using custom exceptions for error handling. Specifically, create an InvalidPinFormatException to be thrown if the PIN format is incorrect, and an IncorrectPinException if the PIN does not match the predefined valid PIN. The system should allow up to 3 attempts to enter the correct PIN, and if the user exceeds this limit, the account should be locked with an appropriate message. Ensure that your program provides feedback on the remaining attempts and handles all exceptions gracefully, displaying relevant error messages as needed.

Sample input and output

Enter your PIN: 12345

Error: PIN must be a 4-digit number.

Remaining attempts: 2

Enter your PIN: 0000

Error: Incorrect PIN. Please try again.

Remaining attempts: 1

Enter your PIN: 1234

Access granted. Welcome to your account.

File processing complete.

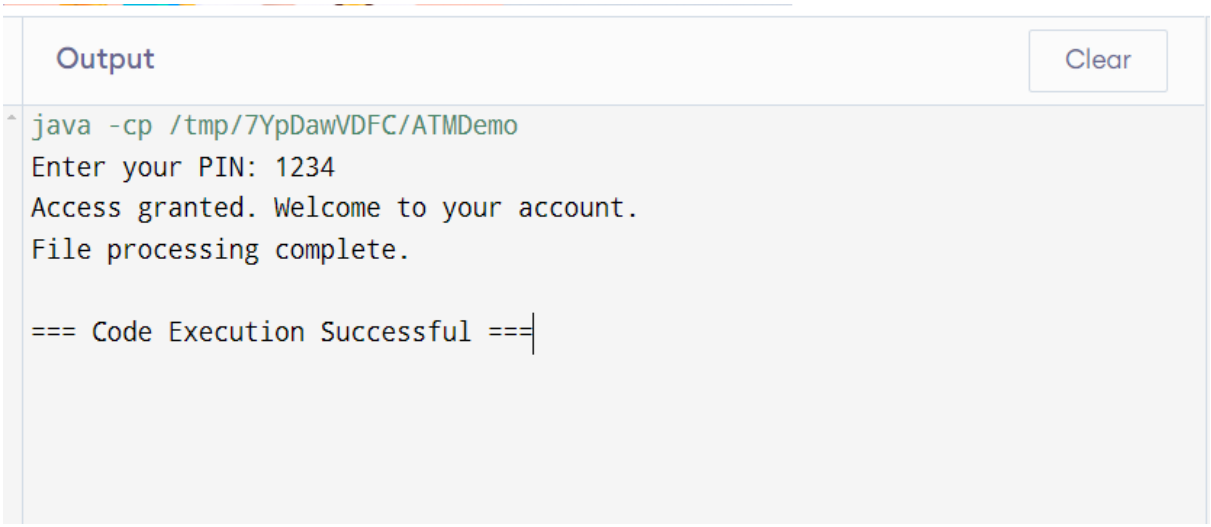
PROGRAM:

```
import java.util.Scanner;

class InvalidPinFormatException extends Exception {
   public InvalidPinFormatException(String message) {
      super(message);
   }
}

class IncorrectPinException extends Exception {
   public IncorrectPinException(String message) {
      super(message);
   }
```

```
}

class ATM {
   private static final String VALID_PIN = "1234";

   private static final int MAX_ATTEMPTS = 3;

   private int attemptsLeft;

   public ATM() {
      this.attemptsLeft = MAX_ATTEMPTS;
   }

   public void validatePin(String enteredPin) throws InvalidPinFormatException,
IncorrectPinException {

      if (enteredPin.length() != 4 || !enteredPin.matches("\\d+")) {
         throw new InvalidPinFormatException("PIN must be a 4-digit number.");
      }

      if (!enteredPin.equals(VALID_PIN)) {
         throw new IncorrectPinException("Incorrect PIN. Please try again.");
      }

      System.out.println("Access granted. Welcome to your account.");
   }

   public void accessAccount() {
      Scanner scanner = new Scanner(System.in);

      while (attemptsLeft > 0) {
         System.out.print("Enter your PIN: ");
         String enteredPin = scanner.nextLine();

         try {
            validatePin(enteredPin);
            break;
         } catch (InvalidPinFormatException | IncorrectPinException e) {

            System.out.println("Error: " + e.getMessage());
            attemptsLeft--;

            if (attemptsLeft > 0) {
               System.out.println("Remaining attempts: " + attemptsLeft);
            } else {
               System.out.println("Your account has been locked due to too many failed attempts.");
```

```
            }
        }
    }

    System.out.println("File processing complete.");
  }
}
public class ATMDemo {
  public static void main(String[] args) {
    ATM atm = new ATM();
    atm.accessAccount();
  }
}
```

OUTPUT:



Output Clear

```
java -cp /tmp/ykGpIdNHPZ/ATMDemo
Enter your PIN: 123
ERROR!
Error: PIN must be a 4-digit number.
Remaining attempts: 2
Enter your PIN: 1233
ERROR!
Error: Incorrect PIN. Please try again.
Remaining attempts: 1
Enter your PIN: 1233
ERROR!
Error: Incorrect PIN. Please try again.
Your account has been locked due to too many failed attempts.
File processing complete.

=== Code Execution Successful ===
```