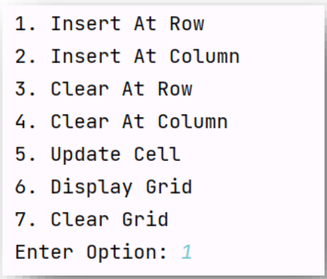
Lab08

Name: Tet Davann

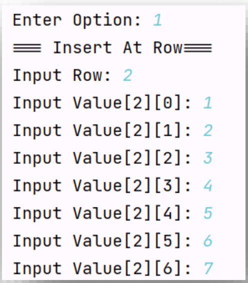
ID: IDTB080023

✓ Lab08.1

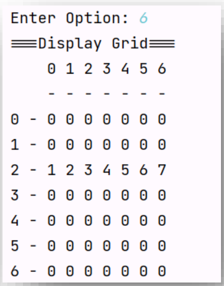
```java
import java.util.Scanner;

interface GridLayout{
    void insertAtRow(int rowNumber,int[] values);
    void insertAtColumn(int columnNumber,int[] values);
    void clearAtRow(int rowNumber);
    void clearAtColumn(int columnNumber);
    void updateCell(int rowNumber,int columnNumber,int values);
    void displayGrid();
    void clear();
}
class Grid implements GridLayout{
    public int[][] table;
    public Grid(int x){
        table =new int[x][x];
    }

    @Override
    public void insertAtRow(int rowNumber, int[] values) {
        for(int i=0;i<table.length;i++){
            table[rowNumber][i] = values[i];
        }
    }
    @Override
    public void insertAtColumn(int columnNumber, int[] values) {
        for(int i = 0; i < table.length; i++) {
            table[i][columnNumber] = values[i];
        }
    }

    @Override
    public void clearAtRow(int rowNumber) {
        insertAtRow(rowNumber,new int[table.length]);
    }

    @Override
    public void clearAtColumn(int columnNumber) {
        insertAtColumn(columnNumber,new int[table.length]);
    }

    @Override
    public void updateCell(int rowNumber, int columnNumber, int values) {
        table[rowNumber][columnNumber] = values;
    }

    @Override
    public void displayGrid() {
        System.out.print("    ");
        for(int i = 0;i<table.length;i++){
            System.out.print(i+" ");
        }
        System.out.println();
        System.out.print("    ");
        for(int i = 0;i<table.length;i++){
            System.out.print("- ");
        }
        System.out.println();
        for(int i=0;i<table.length;i++) {
            System.out.print(i+" - ");
            for(int j=0;j<table.length;j++) {
                System.out.print(table[i][j]+" ");
            }
            System.out.println();
        }
    }

    @Override
    public void clear() {
        table = new int[table.length][table.length];
    }
}
public class Lab08_1 {
    private static Scanner sc = new Scanner(System.in);
    private static Grid grid= new Grid(7);
    private static void menu(){
        System.out.print("1. Insert At Row\n" +
                "2. Insert At Column\n" +
                "3. Clear At Row\n" +
                "4. Clear At Column\n" +
                "5. Update Cell\n" +
                "6. Display Grid\n" +
                "7. Clear Grid\n" +
                "Enter Option: ");
        int opt;
        try {
            opt = sc.nextInt();
            switch (opt) {
                case 1 -> {
                    insertAtRow();
                    menu();
                }
                case 2 -> {
                    insertAtColumn();
                    menu();
                }
                case 3 -> {
                    clearAtRow();
                    menu();
                }
                case 4 -> {
                    clearAtColumn();
                    menu();
                }
                case 5 -> {
                    updateCell();
                    menu();
                }
                case 6 -> {
                    displayGrid();
                    menu();
                }
                case 7 -> {
                    clear();
                    menu();
                }
                default -> {
                    System.out.println("Please input option value from 1 to 7");
                    menu();
                }
            }
        }catch (Exception e) {
            System.out.println("Please enter number");
        }


    }
    private static void insertAtRow(){
        System.out.println("=== Insert At Row===");
        int row=0;
        do{
            System.out.print("Input Row: ");
            try {
                row = sc.nextInt();
                if(row<0){
                    System.out.println("Please enter value a range from 0 to
"+(grid.table.length-1));
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                row =0;
            }
        }while (row>grid.table.length||row<=0);
        int[] values = new int[grid.table.length];
        for(int i=0;i<values.length;i++) {
            System.out.print("Input Value["+row+"]["+i+"]: ");
            values[i] = sc.nextInt();
            if(values[i]>9){
                i=i-1;
                System.out.println("Please input value a range from 0 to 9");
            }
        }
        grid.insertAtRow(row,values);
    }
    private static void insertAtColumn(){
        System.out.println("=== Insert At Column===");
        int column=0;
        do{
            System.out.print("Input Column: ");
            try {
                column = sc.nextInt();
                if(column<=0){
                    System.out.println("Please enter value a range from 0 to
"+(grid.table.length-1));
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                column =0;
            }
        }while (column>grid.table.length||column<=0);
        int[] values = new int[grid.table.length];
        for(int i=0;i<values.length;i++) {
            System.out.print("Input Value["+i+"]["+column+"]: ");
            values[i] = sc.nextInt();
            if(values[i]>9){
                i=i-1;
                System.out.println("Please input value a range from 0 to 9");
            }
        }
        grid.insertAtColumn(column,values);
    }
    private static void clearAtRow(){
        System.out.println("=== Clear At Row===");
        int row=0;
        do{
            System.out.print("Input Row: ");
            try {
                row = sc.nextInt();
                if(row<=0){
                    System.out.println("Please enter value a range from 0 to 
"+(grid.table.length-1));
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                row =0;
            }
        }while (row>grid.table.length||row<=0);
        grid.clearAtRow(row);
        menu();
    }
    private static void clearAtColumn(){
        System.out.println("=== Clear At Column===");
        int column=0;
        do{
            System.out.print("Input Column: ");
            try {
                column = sc.nextInt();
                if(column<=0){
                    System.out.println("Please enter value a range from 0 to 
"+(grid.table.length-1));
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                column =0;
            }
        }while (column>=grid.table.length||column<0);
        grid.clearAtColumn(column);
    }
    private static void updateCell(){
        System.out.println("=== Update Cell===");
        int row=0;
        do{
            System.out.print("Input Row: ");
            try {
                row = sc.nextInt();
                if(row<=0){
                    System.out.println("Please enter value a range from 1 to
"+grid.table.length);
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                row =0;
            }
        }while (row>grid.table.length||row<=0);
        int column=0;
        do{
            System.out.print("Input Column: ");
            try {
                column = sc.nextInt();
                if(column<=0){
                    System.out.println("Please enter value a range from 1 to
"+grid.table.length);
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                column =0;
            }
        }while (column>grid.table.length||column<=0);
        int value=0;
        do{
            System.out.print("Input Value: ");
            try {
                value = sc.nextInt();
                if(value>9||value<=0){
                    System.out.println("Please enter value a range from 0 to 9");
                }
            }catch (Exception e) {
                System.out.print("Please enter number");
                value =0;
            }
        }while (value>9||value<=0);
        grid.updateCell(row,column,value);
    }
    private static void displayGrid(){
        System.out.println("===Display Grid===");
        grid.displayGrid();
    }
    private static void clear(){
        System.out.println("===Clear Grid===");
        grid.clear();
        System.out.println("Grid is cleared successfully");
    }
    public static void main(String[] args){
        menu();
    }
}
```



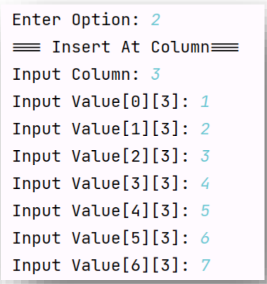
```
1. Insert At Row
2. Insert At Column
3. Clear At Row
4. Clear At Column
5. Update Cell
6. Display Grid
7. Clear Grid
Enter Option: 1
```



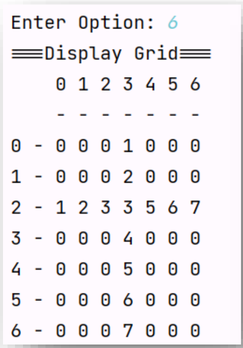
```
Enter Option: 1
═══ Insert At Row═══
Input Row: 2
Input Value[2][0]: 1
Input Value[2][1]: 2
Input Value[2][2]: 3
Input Value[2][3]: 4
Input Value[2][4]: 5
Input Value[2][5]: 6
Input Value[2][6]: 7
```



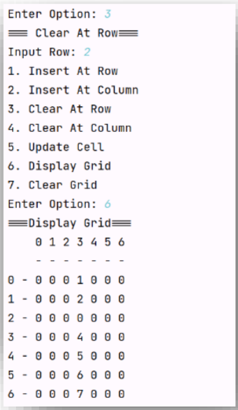
```
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 0 0 0 0
1 - 0 0 0 0 0 0 0
2 - 1 2 3 4 5 6 7
3 - 0 0 0 0 0 0 0
4 - 0 0 0 0 0 0 0
5 - 0 0 0 0 0 0 0
6 - 0 0 0 0 0 0 0
```



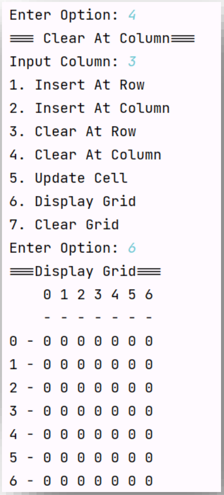
```
Enter Option: 2
═══ Insert At Column═══
Input Column: 3
Input Value[0][3]: 1
Input Value[1][3]: 2
Input Value[2][3]: 3
Input Value[3][3]: 4
Input Value[4][3]: 5
Input Value[5][3]: 6
Input Value[6][3]: 7
```



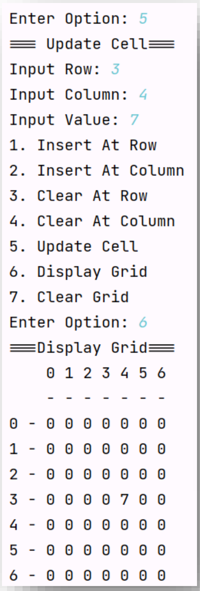
```
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 1 0 0 0
1 - 0 0 0 2 0 0 0
2 - 1 2 3 3 5 6 7
3 - 0 0 0 4 0 0 0
4 - 0 0 0 5 0 0 0
5 - 0 0 0 6 0 0 0
6 - 0 0 0 7 0 0 0
```



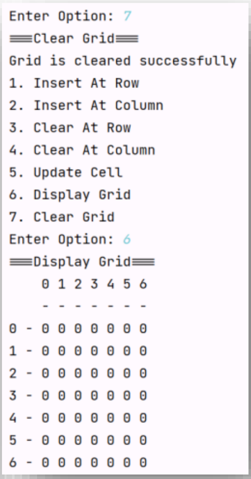
```
Enter Option: 3
═══ Clear At Row═══
Input Row: 2
1. Insert At Row
2. Insert At Column
3. Clear At Row
4. Clear At Column
5. Update Cell
6. Display Grid
7. Clear Grid
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 1 0 0 0
1 - 0 0 0 2 0 0 0
2 - 0 0 0 0 0 0 0
3 - 0 0 0 4 0 0 0
4 - 0 0 0 5 0 0 0
5 - 0 0 0 6 0 0 0
6 - 0 0 0 7 0 0 0
```



```
Enter Option: 4
═══ Clear At Column═══
Input Column: 3
1. Insert At Row
2. Insert At Column
3. Clear At Row
4. Clear At Column
5. Update Cell
6. Display Grid
7. Clear Grid
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 0 0 0 0
1 - 0 0 0 0 0 0 0
2 - 0 0 0 0 0 0 0
3 - 0 0 0 0 0 0 0
4 - 0 0 0 0 0 0 0
5 - 0 0 0 0 0 0 0
6 - 0 0 0 0 0 0 0
```



```
Enter Option: 5
═══ Update Cell═══
Input Row: 3
Input Column: 4
Input Value: 7
1. Insert At Row
2. Insert At Column
3. Clear At Row
4. Clear At Column
5. Update Cell
6. Display Grid
7. Clear Grid
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 0 0 0 0
1 - 0 0 0 0 0 0 0
2 - 0 0 0 0 0 0 0
3 - 0 0 0 0 7 0 0
4 - 0 0 0 0 0 0 0
5 - 0 0 0 0 0 0 0
6 - 0 0 0 0 0 0 0
```

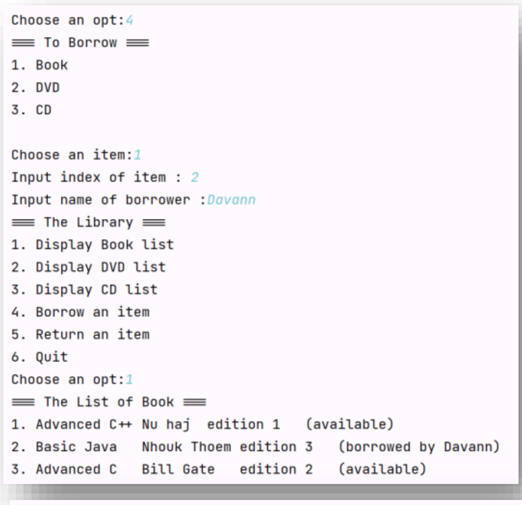


```
Enter Option: 7
═══Clear Grid═══
Grid is cleared successfully
1. Insert At Row
2. Insert At Column
3. Clear At Row
4. Clear At Column
5. Update Cell
6. Display Grid
7. Clear Grid
Enter Option: 6
═══Display Grid═══
    0 1 2 3 4 5 6
    - - - - - - -
0 - 0 0 0 0 0 0 0
1 - 0 0 0 0 0 0 0
2 - 0 0 0 0 0 0 0
3 - 0 0 0 0 0 0 0
4 - 0 0 0 0 0 0 0
5 - 0 0 0 0 0 0 0
6 - 0 0 0 0 0 0 0
```

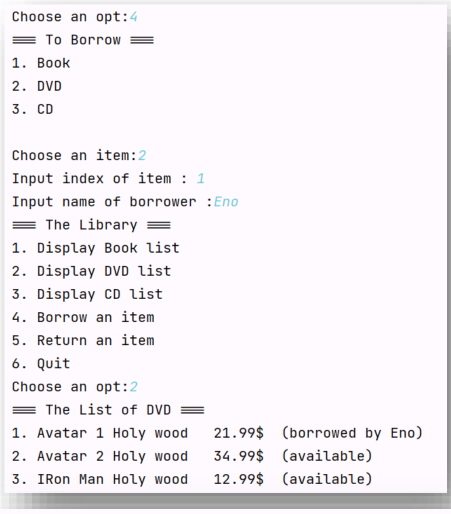
✓ Lab08.2

```java
import java.util.Scanner;

interface Borrowable{
    void borrow(String borrower);
    void returnItem();
    boolean isBorrowed();
}
class BorrowableItem{
    private String borrower;
    private boolean borrowed;
    public String getBorrower() {
        return borrower;
    }

    public void setBorrower(String borrower) {
        this.borrower = borrower;
    }

    public boolean isBorrowed() {
        return borrowed;
    }
    public void setBorrowed(boolean borrowed) {
        this.borrowed = borrowed;
    }


}
class Book extends BorrowableItem implements Borrowable{
    private String title;
    private String edition;
    private String author;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getEdition() {
        return edition;
    }

    public void setEdition(String edition) {
        this.edition = edition;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    @Override
    public void borrow(String borrower) {
        setBorrower(borrower);
    }

    @Override
    public void returnItem() {
        setBorrowed(false);
    }


}
class DVD extends BorrowableItem implements Borrowable{
    private String price;
    private String category;
    private String title;
    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }


    @Override
    public void borrow(String borrower) {
        setBorrower(borrower);
    }

    @Override
    public void returnItem() {
        setBorrowed(false);
    }

}
class CD extends BorrowableItem implements Borrowable{
    private String price;
    private String singer;
    private String category;

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public String getSinger() {
        return singer;
    }

    public void setSinger(String singer) {
        this.singer = singer;
    }
    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    @Override
    public void borrow(String borrower) {
        setBorrower(borrower);
    }

    @Override
    public void returnItem() {
        setBorrowed(false);
    }


}
public class Lab08_2 {
    private static String[][] book = {{"Advanced C++","Nu haj","edition
1"},{"Basic Java","Nhouk Thoem","edition 3"},{ "Advanced C","Bill Gate","edition
2"}};
    private static String[][] dvd ={{"Avatar 1","Holy wood","21.99$"},{"Avatar
2","Holy wood","34.99$"},{"IRon Man","Holy wood","12.99$"}};
    private static String[][] cd ={{"KPOP","Plack
Pink","9.99$"},{"KPOP","BTS","4.99$"},{"Traditional Music","Koev
Sarat","19.99$"}};
    private static Book[] objBook = new Book[book.length];
    private static DVD[] objDVD = new DVD[dvd.length];
    private static CD[] objCD = new CD[cd.length];
    private static Scanner sc= new Scanner(System.in);
    private static void initValues() {
        for(int i=0;i<3;i++){
            objBook[i] = new Book();
            objDVD[i] = new DVD();
            objCD[i] = new CD();
            //book
            objBook[i].setTitle(book[i][0]);
            objBook[i].setAuthor(book[i][1]);
            objBook[i].setEdition(book[i][2]);
            objBook[i].returnItem();
            //dvd
            objDVD[i].setTitle(dvd[i][0]);
            objDVD[i].setCategory(dvd[i][1]);
            objDVD[i].setPrice(dvd[i][2]);
            objDVD[i].returnItem();
            //cd
            objCD[i].setCategory(cd[i][0]);
            objCD[i].setSinger(cd[i][1]);
            objCD[i].setPrice(cd[i][2]);
            objCD[i].returnItem();
        }
    }
    private static void menu(){
        System.out.print("=== The Library ===\n" +
                "1. Display Book list\n" +
                "2. Display DVD list\n" +
                "3. Display CD list\n" +
                "4. Borrow an item\n" +
                "5. Return an item\n" +
                "6. Quit\n" +
                "Choose an opt:");
        int opt = sc.nextInt();
        switch (opt) {
            case 1 -> {
                System.out.println("=== The List of Book ===");
                listBook();
                menu();
            }
            case 2 -> {
                System.out.println("=== The List of DVD ===");
                listDVD();
                menu();
            }
            case 3 -> {
                System.out.println("=== The List of CD ===");
                listCD();
                menu();
            }
            case 4 -> {
                System.out.println("=== To Borrow ===");
                borrowItem();
                menu();
            }
            case 5 -> {
                System.out.println("=== To Return Item ===");
                returnItem();
                menu();
            }
            case 6 -> {
                System.out.println("Quited");
            }
            default -> menu();
        }
    }
    private static void listBook(){

        for(int i=0;i< objBook.length;i++){
            if(objBook[i].isBorrowed()){
                System.out.println((i+1)+". "+ objBook[i].getTitle()+"\t"+
objBook[i].getAuthor()+"\t"+ objBook[i].getEdition()+"\t(borrowed by
"+objBook[i].getBorrower()+")");

            }else {
                System.out.println((i+1)+". "+ objBook[i].getTitle()+"\t"+
objBook[i].getAuthor()+"\t"+ objBook[i].getEdition()+"\t(available)");
            }
        }
    }
    private static void listDVD(){

        for(int i=0;i< objDVD.length;i++){
            if(objDVD[i].isBorrowed()){
                System.out.println((i+1)+". "+ objDVD[i].getTitle()+"\t"+
objDVD[i].getCategory()+"\t"+ objDVD[i].getPrice()+"\t(borrowed by
"+objDVD[i].getBorrower()+")");

            }else {
                System.out.println((i+1)+". "+ objDVD[i].getTitle()+"\t"+
objDVD[i].getCategory()+"\t"+ objDVD[i].getPrice()+"\t(available)");
            }
        }
    }
    private static void listCD(){
        for(int i=0;i< objCD.length;i++){
            if(objCD[i].isBorrowed()){
                System.out.println((i+1)+". "+ objCD[i].getCategory()+"\t"+
objCD[i].getSinger()+"\t"+ objCD[i].getPrice()+"\t(borrowed by
"+objCD[i].getBorrower()+")");

            }else {
                System.out.println((i+1)+". "+ objCD[i].getCategory()+"\t"+
objCD[i].getSinger()+"\t"+ objCD[i].getPrice()+"\t(available)");
            }
        }
    }
    private static void borrowItem(){
        System.out.print("1. Book\n" +
                "2. DVD\n" +
                "3. CD\n\n" +
                "Choose an item:");
        int opt = sc.nextInt();
        int index;
        String name;
        switch (opt) {
            case 1->{
                System.out.print("Input index of item : ");
                index = sc.nextInt();
                if(!objBook[index-1].isBorrowed()){
                    System.out.print("Input name of borrower :" );
                    name = sc.next();
                    objBook[index-1].setBorrower(name);
                    objBook[index-1].setBorrowed(true);
                }else{
                    System.out.println("not allow");
                }
            }
            case 2->{
                System.out.print("Input index of item : ");
```

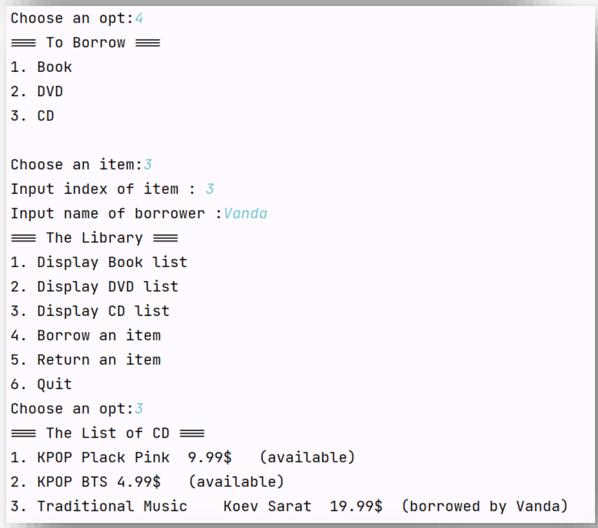
```java
                index = sc.nextInt();
                if(!objDVD[index-1].isBorrowed()){
                    System.out.print("Input name of borrower :" );
                    name = sc.next();
                    objDVD[index-1].setBorrower(name);
                    objDVD[index-1].setBorrowed(true);
                }else{
                    System.out.println("not allow");
                }
            }
            case 3->{
                System.out.print("Input index of item : ");
                index = sc.nextInt();
                if(!objCD[index-1].isBorrowed()){
                    System.out.print("Input name of borrower :" );
                    name = sc.next();
                    objCD[index-1].setBorrower(name);
                    objCD[index-1].setBorrowed(true);
                }else{
                    System.out.println("not allow");
                }
            }
            default -> borrowItem();
        }
    }
    private static void returnItem(){
        System.out.print("1. Book\n" +
                "2. DVD\n" +
                "3. CD\n\n" +
                "Choose an item:");
        int opt = sc.nextInt();
        int index;
        System.out.print("Input index of item : ");
        index = sc.nextInt();
        switch (opt) {
            case 1->{
                if(objBook[index-1].isBorrowed()){
                    objBook[index-1].returnItem();
                }else{
                    System.out.println("not allow");
                }
            }
            case 2->{
                if(objDVD[index-1].isBorrowed()){
                    objDVD[index-1].returnItem();
                }else{
                    System.out.println("not allow");
                }
            }
            case 3->{
                if(objCD[index-1].isBorrowed()){
                    objCD[index-1].returnItem();
                }else{
                    System.out.println("not allow");
                    }
                }
                default -> returnItem();
            }
        }
    public static void main(String[] args){
        initValues();
        menu();
    }
}
```



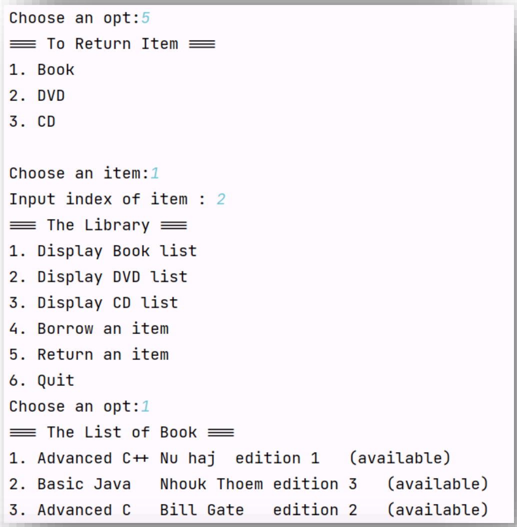
```
Choose an opt:4
== To Borrow ==
1. Book
2. DVD
3. CD

Choose an item:1
Input index of item : 2
Input name of borrower :Davann
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:1
== The List of Book ==
1. Advanced C++ Nu haj  edition 1   (available)
2. Basic Java   Nhouk Thoem edition 3   (borrowed by Davann)
3. Advanced C   Bill Gate   edition 2   (available)
```



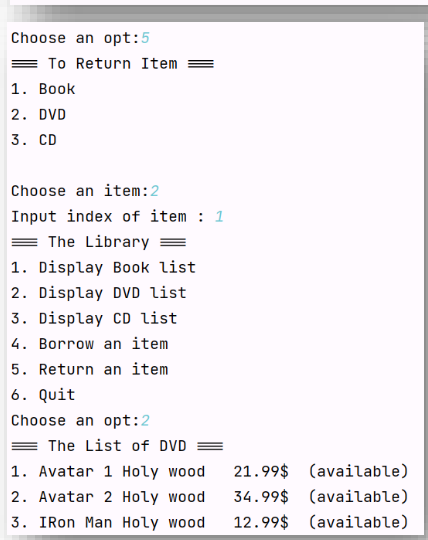
```
Choose an opt:4
== To Borrow ==
1. Book
2. DVD
3. CD

Choose an item:2
Input index of item : 1
Input name of borrower :Eno
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:2
== The List of DVD ==
1. Avatar 1 Holy wood   21.99$  (borrowed by Eno)
2. Avatar 2 Holy wood   34.99$  (available)
3. IRon Man Holy wood   12.99$  (available)
```



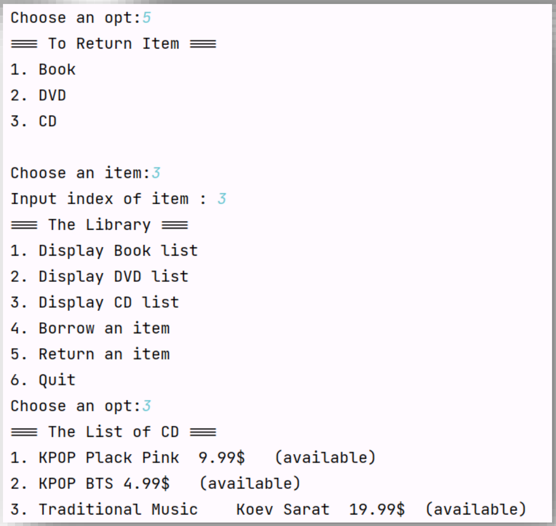
```
Choose an opt:4
== To Borrow ==
1. Book
2. DVD
3. CD

Choose an item:3
Input index of item : 3
Input name of borrower :Vanda
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:3
== The List of CD ==
1. KPOP Plack Pink  9.99$   (available)
2. KPOP BTS 4.99$   (available)
3. Traditional Music    Koev Sarat  19.99$  (borrowed by Vanda)
```



```
Choose an opt:5
== To Return Item ==
1. Book
2. DVD
3. CD

Choose an item:1
Input index of item : 2
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:1
== The List of Book ==
1. Advanced C++ Nu haj  edition 1   (available)
2. Basic Java   Nhouk Thoem edition 3   (available)
3. Advanced C   Bill Gate   edition 2   (available)
```



```
Choose an opt:5
== To Return Item ==
1. Book
2. DVD
3. CD

Choose an item:2
Input index of item : 1
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:2
== The List of DVD ==
1. Avatar 1 Holy wood   21.99$  (available)
2. Avatar 2 Holy wood   34.99$  (available)
3. IRon Man Holy wood   12.99$  (available)
```

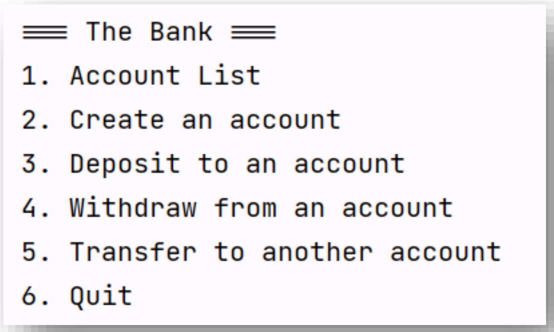


```
Choose an opt:5
== To Return Item ==
1. Book
2. DVD
3. CD

Choose an item:3
Input index of item : 3
== The Library ==
1. Display Book list
2. Display DVD list
3. Display CD list
4. Borrow an item
5. Return an item
6. Quit
Choose an opt:3
== The List of CD ==
1. KPOP Plack Pink  9.99$   (available)
2. KPOP BTS 4.99$   (available)
3. Traditional Music    Koev Sarat  19.99$  (available)
```

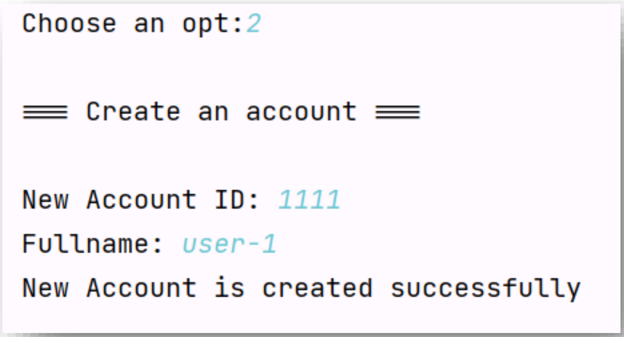
✓ Lab08.3

```java
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

class Account{
    private String id;
    private String fullname;
    private double balance;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getFullname() {
        return fullname;
    }

    public void setFullname(String fullname) {
        this.fullname = fullname;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

}
public class Lab08_3 {
    private static Scanner sc = new Scanner(System.in);
    private static ArrayList<Account> accounts = new ArrayList<>();
    private static void menu(){
        System.out.print("\n=== The Bank ===\n" +
                "1. Account List\n" +
                "2. Create an account\n" +
                "3. Deposit to an account\n" +
                "4. Withdraw from an account\n" +
                "5. Transfer to another account\n" +
                "6. Quit\n" +
                "Choose an opt:");
        int opt = sc.nextInt();
        switch (opt) {
            case 1->{
                System.out.println("\n=== Account List ===");
                listAccount();
                menu();
            }
            case 2->{
                System.out.println("\n=== Create an account ===");
                createAccount();
                menu();
            }
            case 3->{
                System.out.println("\n=== Deposit to an account ===");
                deposit();
                menu();
            }
            case 4->{
                System.out.println("\n=== Withdraw from an account ===");
                withdraw();
                menu();
            }
            case 5->{
                System.out.println("\n=== Transfer to another account ===");
                transfer();
                menu();
            }
            case 6->System.out.println("Quited");
            default -> menu();
        }
    }
    private static void listAccount(){
       if(accounts.size() == 0){
           System.out.println("Account not available");
           return;
       }
       int no=1;
       for(Account account : accounts){
           System.out.println("\nNO #"+no);
           System.out.println("ID: "+account.getId());
           System.out.println("Holder: "+account.getFullname());
           System.out.println("Balance: "+account.getBalance()+"$");
           no++;
       }
    }
    private static void createAccount(){
        System.out.print("\nNew Account ID: ");
        String id = sc.next();
        for(Account account : accounts){
           if(account.getId()==id){
               System.out.println("This id has been already!");
               createAccount();
           }
        }
        System.out.print("Fullname: ");
        sc.nextLine();
        String fullname = sc.nextLine();
        Account account = new Account();
        account.setId(id);
        account.setFullname(fullname);
        account.setBalance(0);
        accounts.add(account);
        System.out.println("New Account is created successfully");
        menu();
    }
    private static void deposit(){
        System.out.print("Account ID: ");
        String id = sc.next();
        boolean verify = false;
        int index =0;
        for(Account account : accounts){
            if(account.getId().equals(id)){
                System.out.println("* "+ account.getFullname());
                verify = true;
                break;
            }
            index++;
        }
        double amount;
        if(verify){
            System.out.print("Deposit amount :");
            try {
                amount = sc.nextDouble();

accounts.get(index).setBalance(accounts.get(index).getBalance()+amount);
                    System.out.println("Deposition is successfully");
            }catch (InputMismatchException e){
                System.out.println("Don't input characters.Please input only
digit");
                deposit();
            }
        }else{
            System.out.println("This account is not available");
            deposit();
        }
    }
    private static void withdraw() {
        System.out.print("Account ID: ");
        String id = sc.next();
        boolean verify = false;
        int index =0;
        for(Account account : accounts){
            if(account.getId().equals(id)){
                System.out.println("* "+ account.getFullname());
                verify = true;
                break;
            }
            index++;
        }
        double amount;
        if(verify){
            System.out.print("Withdraw amount :");
            try {
                amount = sc.nextDouble();
                if(amount<=accounts.get(index).getBalance()){

accounts.get(index).setBalance(accounts.get(index).getBalance()-amount);
                    System.out.println("Withdraw is successfully");
                }else{
                    System.out.println("You have insufficient balance in your
source account");
                    withdraw();
                }
            }catch (InputMismatchException e){
                System.out.println("Don't input characters.Please input only 
digit");
                withdraw();
            }

        }else{
            System.out.println("This account is not available");
            withdraw();
        }
    }
    private static void transfer(){
        System.out.print("From ID: ");
        String fromID = sc.next();
        boolean verify = false;
        int index =0;
        for(Account account : accounts){
            if(account.getId().equals(fromID)){
                System.out.println("* "+ account.getFullname());
                verify = true;
                break;
            }
            index++;
        }

        if(verify){
            System.out.print("From ID: ");
            String toID = sc.next();
            boolean verify1 = false;
            int index1 =0;
            for(Account account : accounts){
                if(account.getId().equals(toID)){
                    System.out.println("=> "+ account.getFullname());
                    verify1 = true;
                    break;
                }
                index1++;
            }
            double amount;
            if(verify1){
                System.out.print("Amount: ");
                try{
                    amount = sc.nextDouble();
                    if(amount<=accounts.get(index).getBalance()){

accounts.get(index).setBalance(accounts.get(index).getBalance()-amount);

accounts.get(index1).setBalance(accounts.get(index1).getBalance()+amount);
                        System.out.println("Transfer is successfully");
                    }else{
                        System.out.println("You have insufficient balance in your 
source account");
                        transfer();
                    }
                }catch (InputMismatchException e){
                    System.out.println("Don't input characters.Please input only 
digit");
```

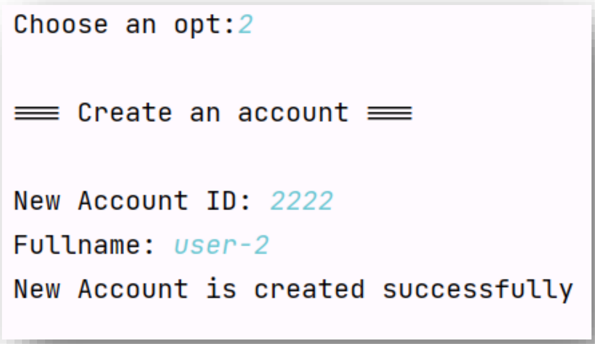
```java
                    transfer();
                }

            }else{
                System.out.println("Thise account not available");
                transfer();
            }
        }else{
            System.out.println("Thise account not available");
            transfer();
        }
    }
    public static void main(String[] args){
        menu();
    }
}
```



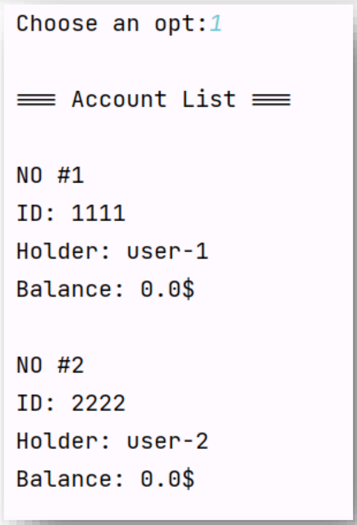
```
=== The Bank ===
1. Account List
2. Create an account
3. Deposit to an account
4. Withdraw from an account
5. Transfer to another account
6. Quit
```



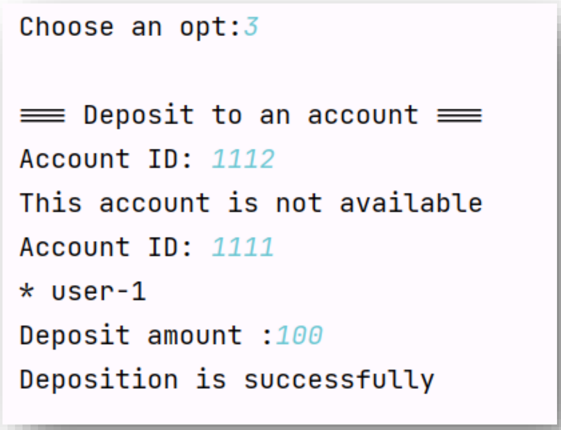
```
Choose an opt:2

=== Create an account ===

New Account ID: 1111
Fullname: user-1
New Account is created successfully
```



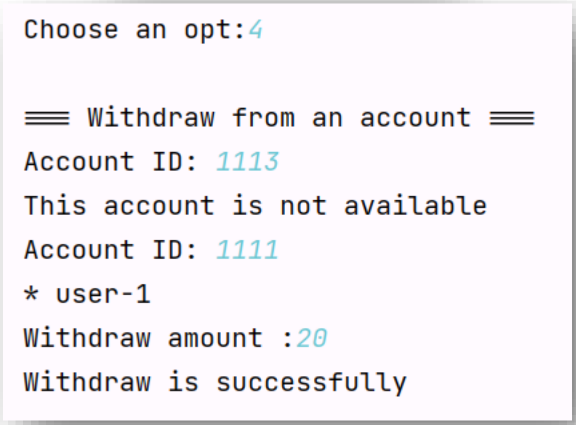
```
Choose an opt:2

=== Create an account ===

New Account ID: 2222
Fullname: user-2
New Account is created successfully
```



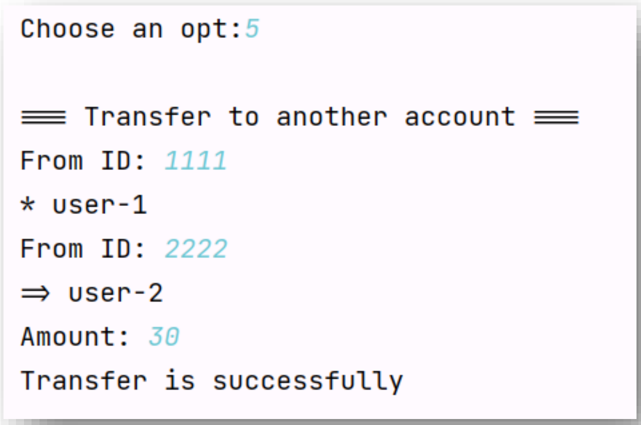
```
Choose an opt:1

=== Account List ===

NO #1
ID: 1111
Holder: user-1
Balance: 0.0$

NO #2
ID: 2222
Holder: user-2
Balance: 0.0$
```



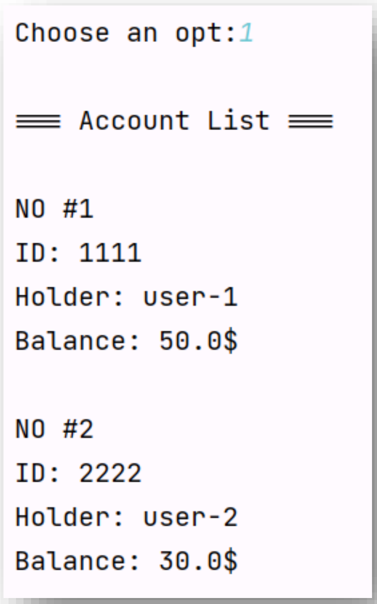
```
Choose an opt:3

=== Deposit to an account ===
Account ID: 1112
This account is not available
Account ID: 1111
* user-1
Deposit amount :100
Deposition is successfully
```



```
Choose an opt:4

=== Withdraw from an account ===
Account ID: 1113
This account is not available
Account ID: 1111
* user-1
Withdraw amount :20
Withdraw is successfully
```



```
Choose an opt:5

═══ Transfer to another account ═══
From ID: 1111
* user-1
From ID: 2222
⇒ user-2
Amount: 30
Transfer is successfully
```



```
Choose an opt:1

═══ Account List ═══

NO #1
ID: 1111
Holder: user-1
Balance: 50.0$

NO #2
ID: 2222
Holder: user-2
Balance: 30.0$
```