A bank manages many customers’ accounts. Each bank account can be a saving or a checking account.

### Bank

- name: String

+ Bank(String name, int size)
+ search(String accNum): boolean
+ add(BankAccount ba): boolean
+ del(String accNum): boolean
+ Setters & Getters

### BankAccount

# accNum: String
# balance: double

+ BankAccount(String accNum, double balance)
+ deposit(double amount)
+ withdraw(double amount): boolean
+ transfer(String accNum, double amount): boolean
+ Setters & Getters

### CheckingAccount

- transactionCount: int

+ CheckingAccount(String accNum, double balance)
+ deposit(double amount): boolean
+ withdraw(double amount): boolean
+ deductfees(): Boolean
+ Setters & Getters

### SavingAccount

- interestRate: double

+ SavingAccount(String accNum, double bal, double interestRate)
+ addInterest() 
+ Setters & Getters

**BankAccount Class.**

The class **BankAccount** provides three services. A method `deposit()` to add an amount to the current balance. A method `withdraw()` to subtract money from the account and ensure that the debit amount does not exceed the account’s balance. A method `transfer()` that deposit money to another account.

**CheckingAccount Class.**

The class **CheckingAccount** should inherit from base class **BankAccount** and include an additional data member `transactionCount` that represents the number of transaction processed. This class should redefine methods `deposit()` and `withdraw()` and implements a new method `deductfees()` that withdraw transactions fees from the account balance whenever the transaction is performed successfully and the number of transaction exceed 3. Such that the fee of one transaction is equal 2 riyals.
SavingAccount Class.

The `savingAccount` class inherits from the `BankAccount` class and includes a new attribute `interestRate` indicating the interest rate assigned to the account. The method `addInterest()` should deposit the amount of interest earned by an account.

Bank Class.

The `Bank` class implements the following methods:

- The constructor `Bank(String Name, int size)` and throws an exception when the array is created with a negative size. (`NegativeArraySizeException`)
- The method `search()` to search an account and returns -1 if it is not founded and its index otherwise.
- The method `add()` to add a new account and throws an exception when an illegal index term in the array has been accessed. (`ArrayIndexOutOfBoundsException`)
- The method `del()` to remove an account and throws an exception when the method has been passed an illegal or inappropriate argument. (`IllegalArgumentException`)

Question
Implement the UML diagram presented previously and write a main program that processes a Bank object according to the following menu. Such that when user tapes 1, the program should display another sub menu concerning only account operations. Throw an exception when the user enters a choice different of the type Integer. (`InputMismatchException`)

```
MENU
-------------------------------
1- Bank Account Operations
5- Print Bank Accounts
6- Exit
-------------------------------
Enter your choice:
```

```
Bank Account Operations
-------------------------------
1- Add a Bank Account
2- Delete a Bank Account
3- Deposit money
4- Withdraw money
5- Deduct transactions fees
6- Add Interest
-------------------------------
Enter your choice:
```