NAME OF PROJECT
Finance Manager

DATE
20/6/2024

PERSONAL FINANCE MANAGER

NAME OF GROUP
CodeHub

PRESENTED BY
CodeHub

NAME OF PROJECT
Finance Manager

NAME OF GROUP
Codehub

SYNOPSIS

Our project, the Finance Manager Project, aimed to assist users in efficiently managing their personal accounts.

The system has features for tracking transactions, savings, bank accounts, budgeting, and report generation. Multiple accounts can be created and managed, spending can be tracked, budget limitations can be set, and savings targets can be kept an eye on. The method facilitates the division of transactions into distinct categories, such as groceries and shopping, which enables thorough financial analysis.

The system's design makes it easy for users to navigate between its various functionalities, making it user-friendly.

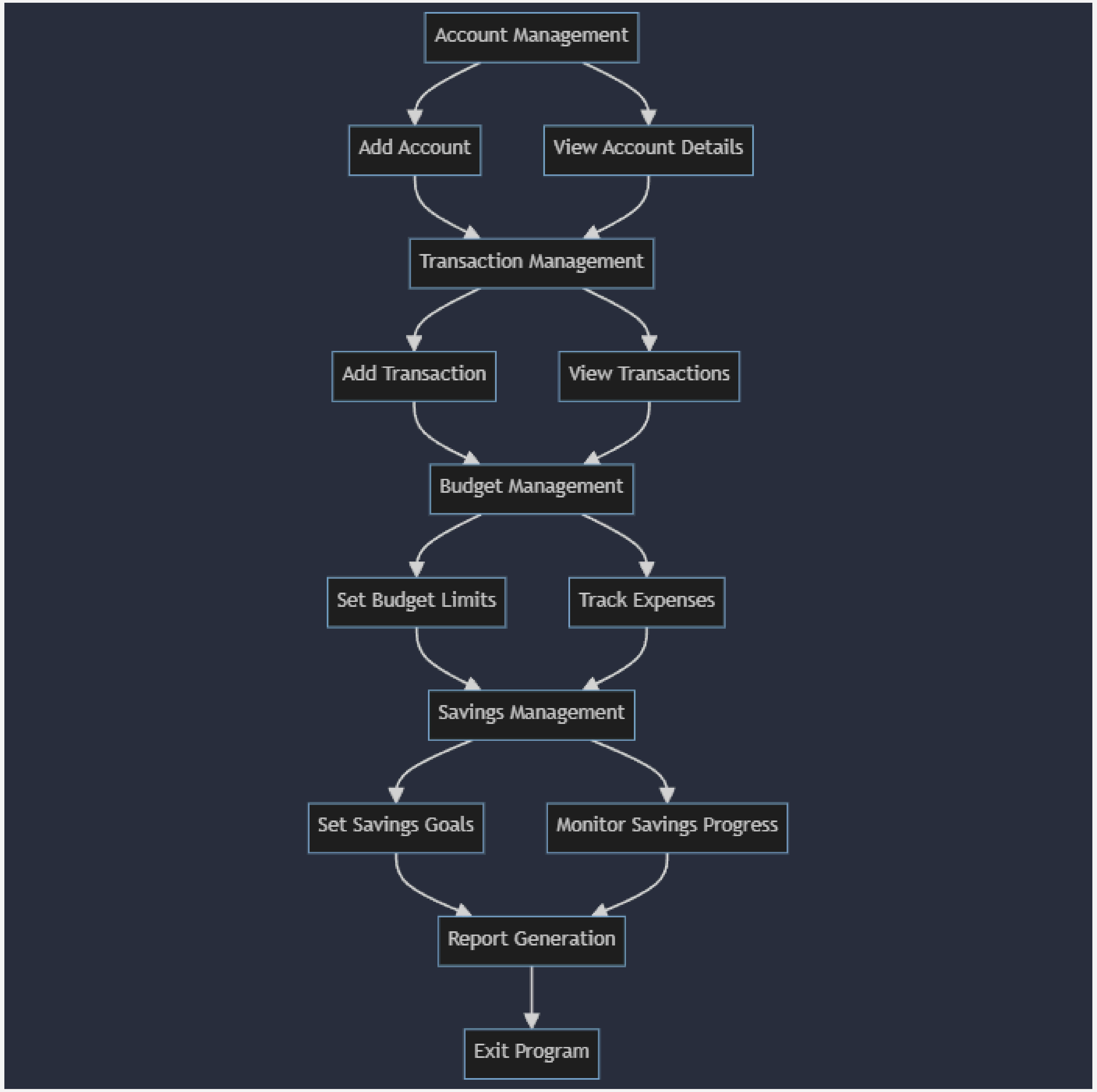
PROJECT SCOPE

The primary objective of the project is to provide users with a user friendly system to manage their finances. The scope are:

- Manage multiple bank accounts simultaneously
- Track and categorize transactions
- Setting and monitoring budget
- Create savings goals
- Analyze financial data based on generated reports
- Protected users accounts data

MAIN COMPONENT

- Account - Responsible for managing user accounts.
- Transaction - Handles deposit and withdrawal processes.
- Budget - Manages user budget allocations.
- Savings - Tracks progress towards savings goals.
- Report - Displays comprehensive details


Account Management
Add Account
View Account Details
Transaction Management
Add Transaction
View Transactions
Budget Management
Set Budget Limits
Track Expenses
Savings Management
Set Savings Goals
Monitor Savings Progress
Report Generation
Exit Program

THE FLOW

MORE DETAILS

Restriction

04

USER

- User can have many account

05

BUDGET

- One Budget per Category.

06

CATEGORY

- Each Transaction is associated with a Category.

07

ACCOUNT

- Belongs to one Bank.
- Can have multiple Transactions.
- Can have multiple Savings.

08

TRANSACTION

- Must be less than the account balance.
- Generates a confirmation message if exceeding budget.

09

SAVING

- Manage by account

10

BANK

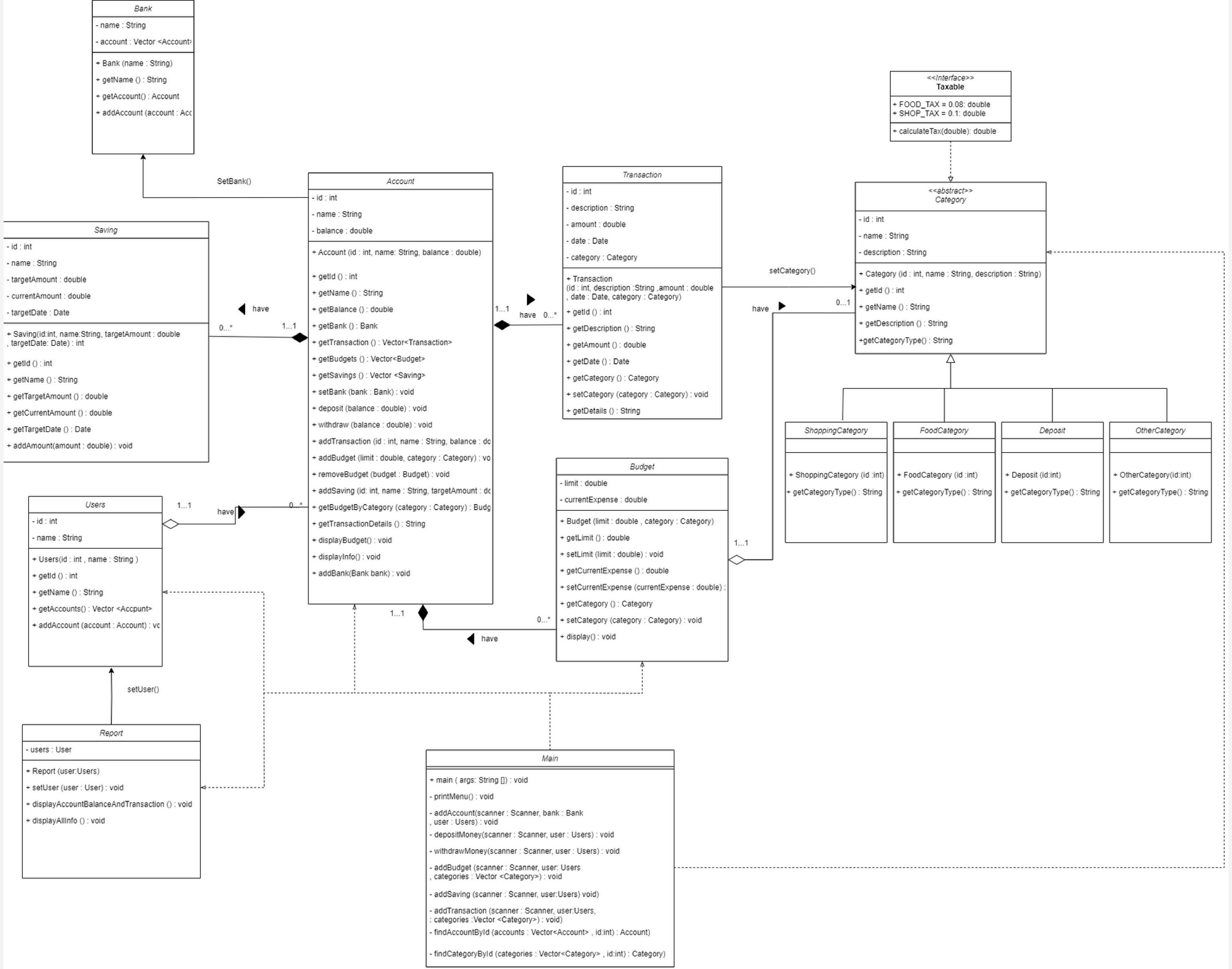
- Each Account is linked to one Bank.

11

EXCEPTION HANDLING

- Insufficient Balance: Throws an exception if a transaction amount is greater than the account balance.
- Exceeding Budget: Throws a warning or confirmation message if a transaction exceeds the set budget for a category.
- Invalid Category: Throws an exception if a non-existent category is referenced.

C L A S S D I A G R A M


Bank
- name : String
- account : Vector <Account>
+ Bank (name : String)
+ getName () : String
+ getAccount() : Account
+ addAccount (account : Acc
SetBank()
Account
- id : int
- name : String
- balance : double
+ Account (id : int, name: String, balance : double)
+ getId () : int
+ getName () : String
+ getBalance () : double
+ getBank () : Bank
+ getTransaction () : Vector<Transaction>
+ getBudgets () : Vector<Budget>
+ getSavings () : Vector <Saving>
+ setBank (bank : Bank) : void
+ deposit (balance : double) : void
+ withdraw (balance : double) : void
+ addTransaction (id : int, name : String, balance : do
+ addBudget (limit : double, category : Category) : vo
+ removeBudget (budget : Budget) : void
+ addSaving (id: int, name : String, targetAmount : do
+ getBudgetByCategory (category : Category) : Budg
+ getTransactionDetails () : String
+ displayBudget() : void
+ displayInfo() : void
+ addBank(Bank bank) : void
Transaction
- id : int
- description : String
- amount : double
- date : Date
- category : Category
+ Transaction (id : int, description :String ,amount : double , date : Date, category : Category)
+ getId () : int
+ getDescription () : String
+ getAmount () : double
+ getDate () : Date
+ getCategory () : Category
+ setCategory (category : Category) : void
+ getDetails () : String
<<Interface>> Taxable
+ FOOD_TAX = 0.08: double
+ SHOP_TAX = 0.1: double
+ calculateTax(double): double
<<abstract>> Category
- id : int
- name : String
- description : String
+ Category (id : int, name : String, description : String)
+ getId () : int
+ getName () : String
+ getDescription () : String
+getCategoryType() : String
setCategory()
ShoppingCategory
+ ShoppingCategory (id :int)
+ getCategoryType() : String
FoodCategory
+ FoodCategory (id :int)
+ getCategoryType() : String
Deposit
+ Deposit (id:int)
+ getCategoryType() : String
OtherCategory
+ OtherCategory(id:int)
+ getCategoryType() : String
Saving
- id : int
- name : String
- targetAmount : double
- currentAmount : double
- targetDate : Date
+ Saving(id:int, name:String, targetAmount : double , targetDate: Date) : int
+ getId () : int
+ getName () : String
+ getTargetAmount () : double
+ getCurrentAmount () : double
+ getTargetDate () : Date
+ addAmount(amount : double) : void
have
0...*
1...1
1...1
have
0...*
have
0...1
Budget
- limit : double
- currentExpense : double
+ Budget (limit : double , category : Category)
+ getLimit () : double
+ setLimit (limit : double) : void
+ getCurrentExpense () : double
+ setCurrentExpense (currentExpense : double) :
+ getCategory () : Category
+ setCategory (category : Category) : void
+ display() : void
1...1
Users
- id : int
- name : String
+ Users(id : int , name : String)
+ getId () : int
+ getName () : String
+ getAccounts() : Vector <Accpunt>
+ addAccount (account : Account) : vo
1...1
have
0...*
1...1
0...*
have
setUser()
Report
- users : User
+ Report (user:Users)
+ setUser (user : User) : void
+ displayAccountBalanceAndTransaction () : void
+ displayAllInfo () : void
Main
+ main (args: String []) : void
- printMenu() : void
- addAccount(scanner : Scanner, bank : Bank , user : Users) : void
- depositMoney(scanner : Scanner, user : Users) : void
- withdrawMoney(scanner : Scanner, user : Users) : void
- addBudget (scanner : Scanner, user: Users , categories : Vector <Category>) : void
- addSaving (scanner : Scanner, user:Users) void)
- addTransaction (scanner : Scanner, user:Users, : categories :Vector <Category>) : void)
- findAccountById (accounts : Vector<Account> , id:int) : Account)
- findCategoryById (categories : Vector<Category> , id:int) : Category)

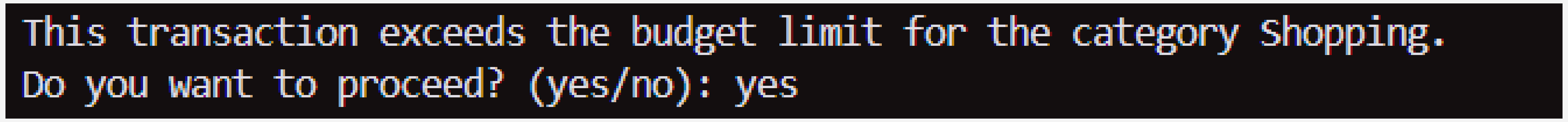
NAME OF PROJECT
Finance Manager

DATE
20/6/2024

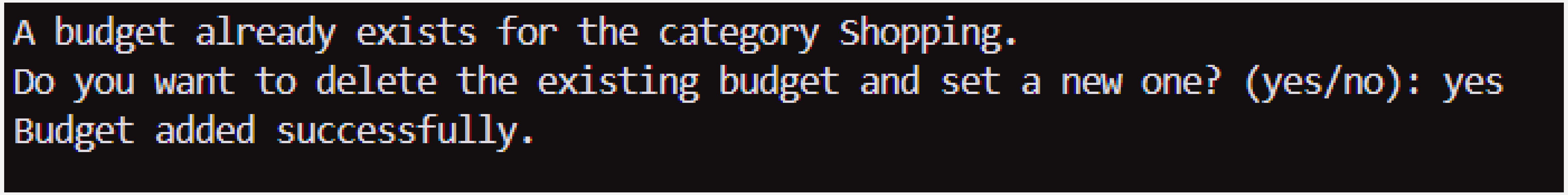
BUDGET

06

If the category has already has budget set and if user want to set the new budget for the category the confirmation message will be display



```
This transaction exceeds the budget limit for the category Shopping.
Do you want to proceed? (yes/no): yes
```



```
A budget already exists for the category Shopping.
Do you want to delete the existing budget and set a new one? (yes/no): yes
Budget added successfully.
```

NAME OF PROJECT
Finance Manager

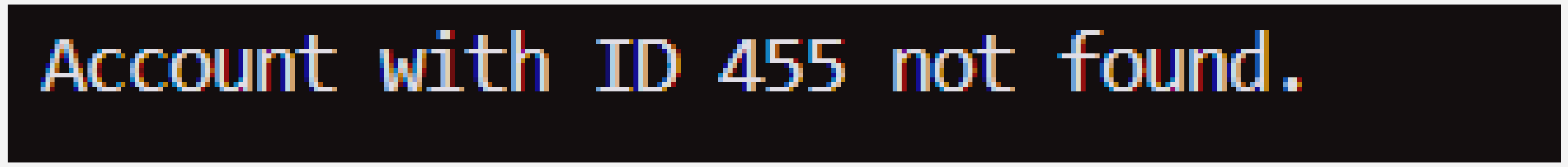
ERROR MESSAGE

DATE
20/6/2024

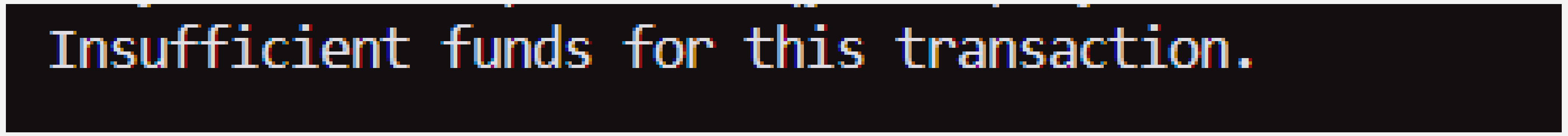
09

The exception handling will display this error message if

- Cannot find account
- Account balance is less than transaction made



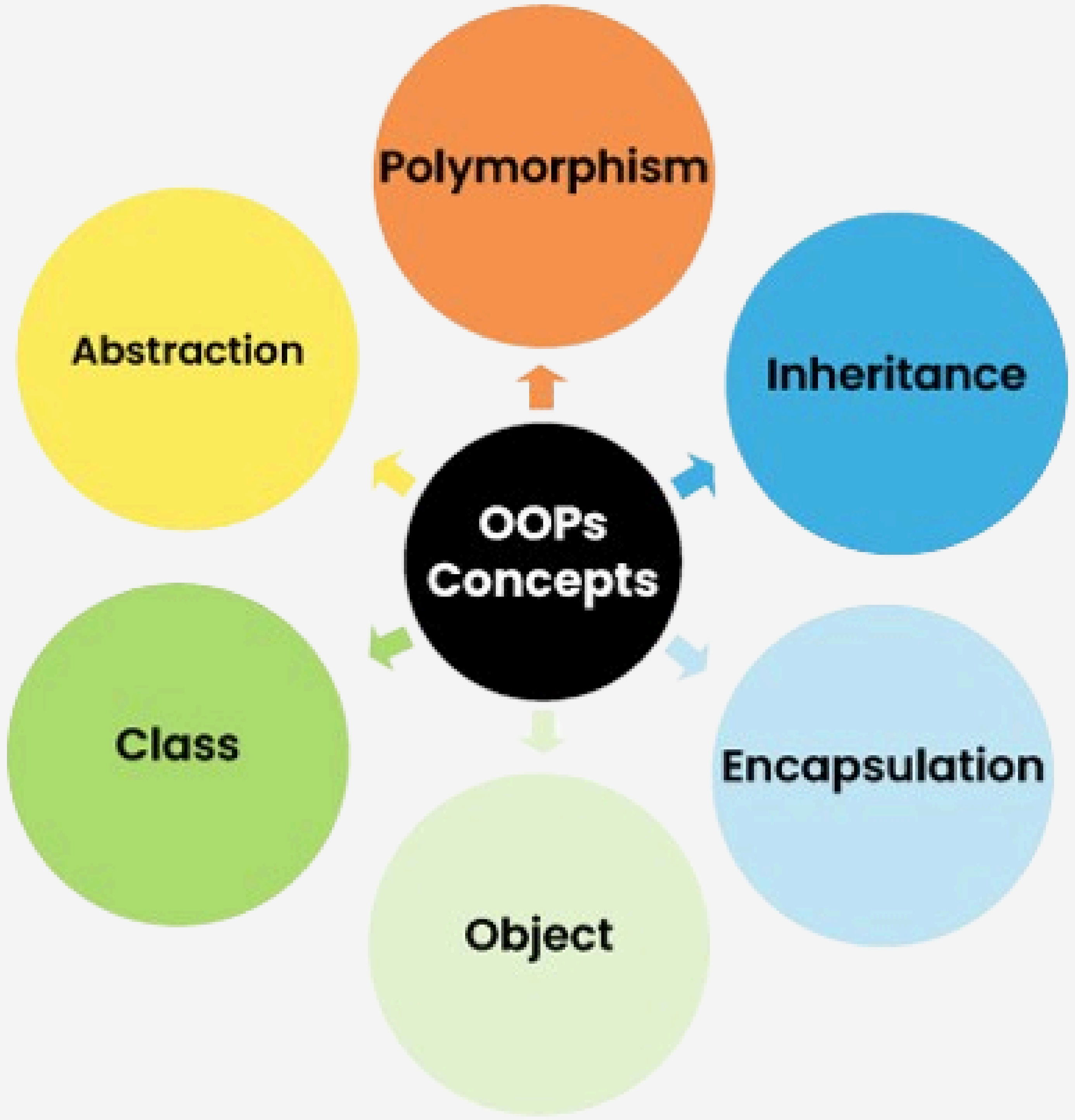
```
Account with ID 455 not found.
```



```
Insufficient funds for this transaction.
```

NAME OF PROJECT
Finance Manager

OOP CONCEPT



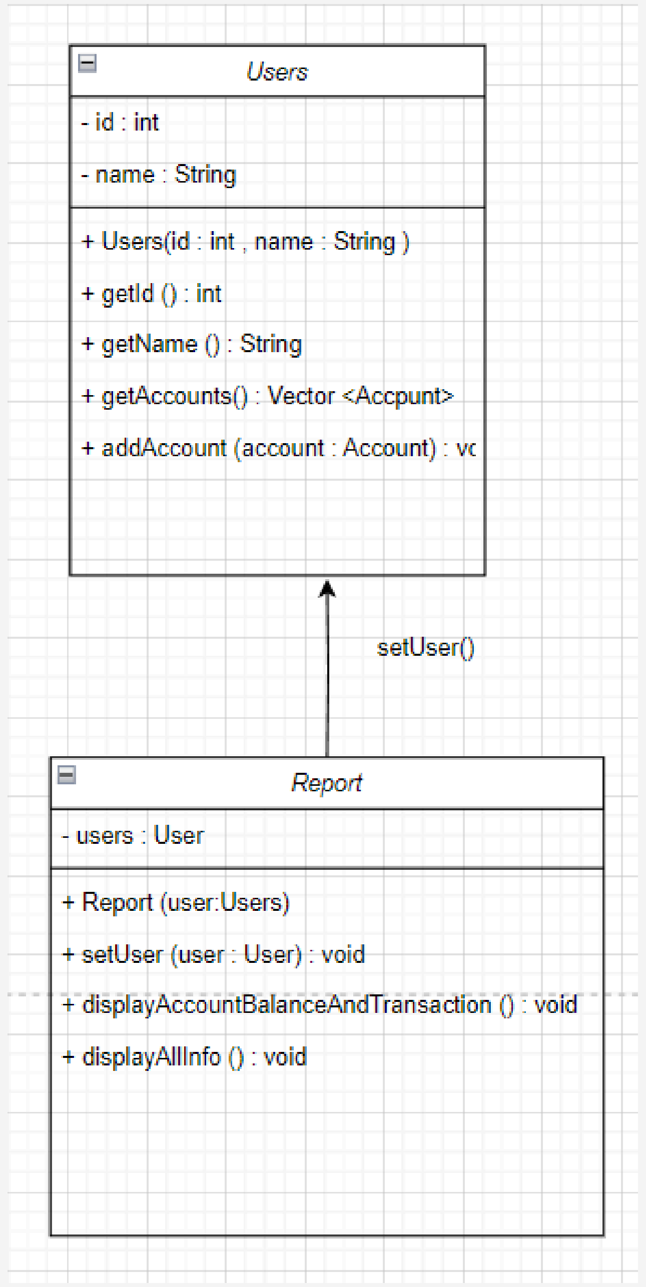
Object-oriented programming has several advantages over procedural programming:

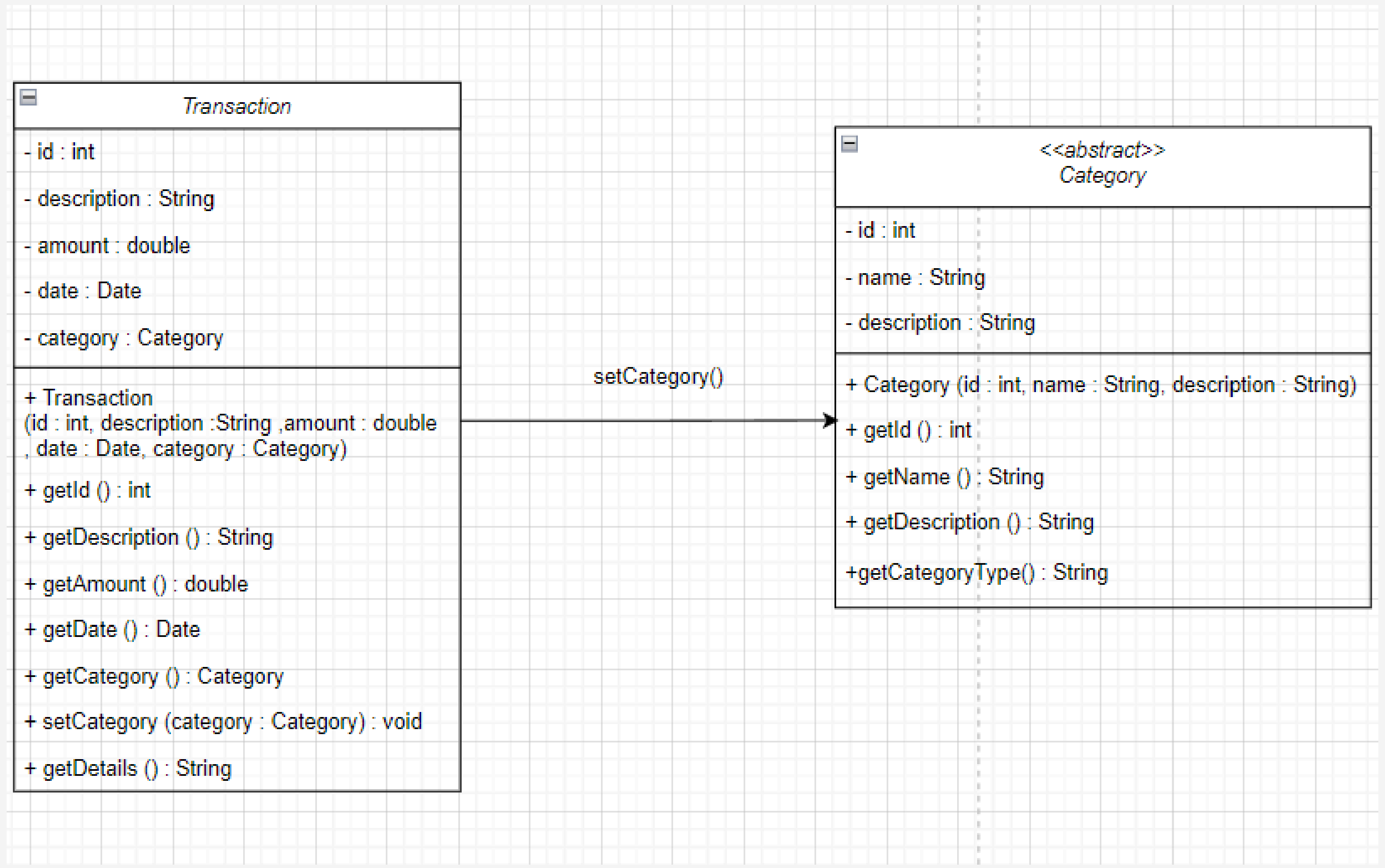
- **OOP is faster and easier to execute**
- **OOP provides a clear structure for the programs**
- **OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug**
- **OOP makes it possible to create full reusable applications with less code and shorter development time**

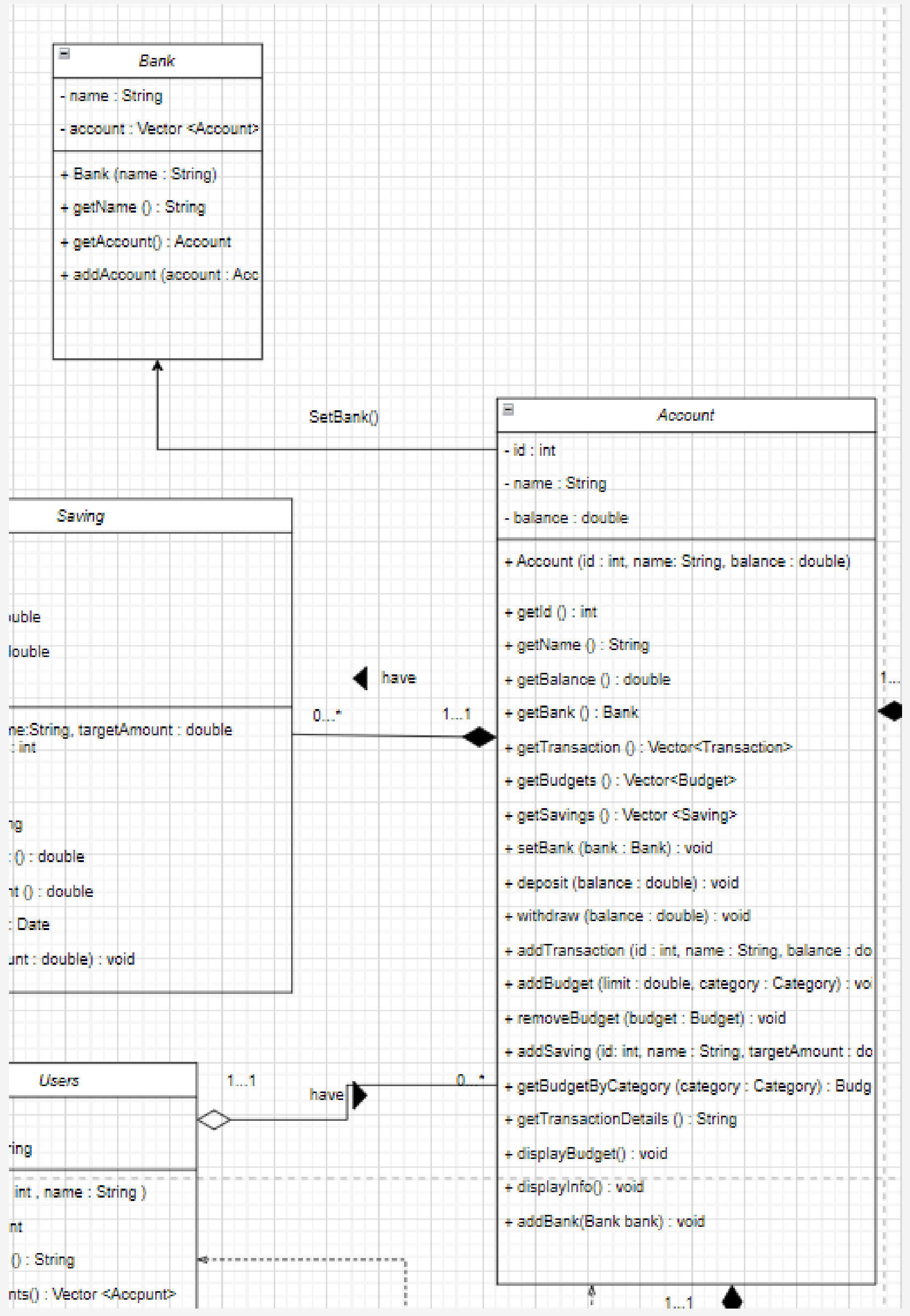
NAME OF PROJECT
Finance Manager

ASSOCIATION

- Association between class User ands Report
- Association between class Bank and Account
- Association between class Transaction and Category


Users
- id : int
- name : String
+ Users(id : int , name : String)
+ getId () : int
+ getName () : String
+ getAccounts() : Vector <Accpunt>
+ addAccount (account : Account) : vo
setUser()
Report
- users : User
+ Report (user:Users)
+ setUser (user : User) : void
+ displayAccountBalanceAndTransaction () : void
+ displayAllInfo () : void

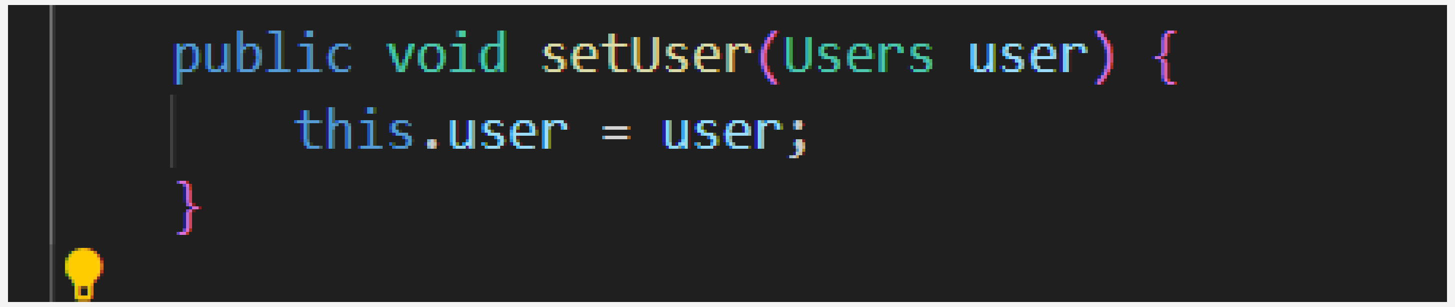



Bank
- name : String
- account : Vector <Account>
+ Bank (name : String)
+ getName () : String
+ getAccount() : Account
+ addAccount (account : Acc
SetBank()
Account
- id : int
- name : String
- balance : double
+ Account (id : int, name: String, balance : double)
+ getId () : int
+ getName () : String
+ getBalance () : double
+ getBank () : Bank
+ getTransaction () : Vector<Transaction>
+ getBudgets () : Vector<Budget>
+ getSavings () : Vector <Saving>
+ setBank (bank : Bank) : void
+ deposit (balance : double) : void
+ withdraw (balance : double) : void
+ addTransaction (id : int, name : String, balance : do
+ addBudget (limit : double, category : Category) : voi
+ removeBudget (budget : Budget) : void
+ addSaving (id: int, name : String, targetAmount : do
+ getBudgetByCategory (category : Category) : Budg
+ getTransactionDetails () : String
+ displayBudget() : void
+ displayInfo() : void
+ addBank(Bank bank) : void
Saving
have
0...*
1...1
Users
1...1
have
0...*
1...1

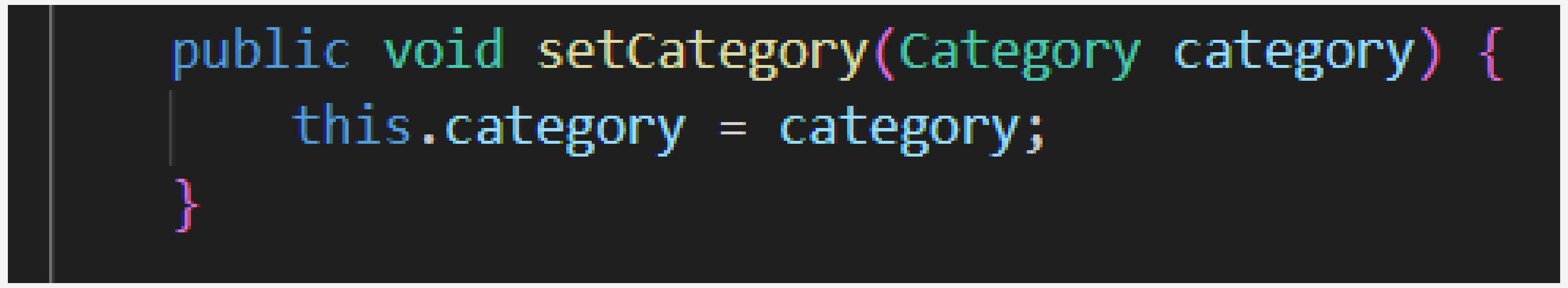
NAME OF PROJECT
Finance Manager

ASSOCIATION

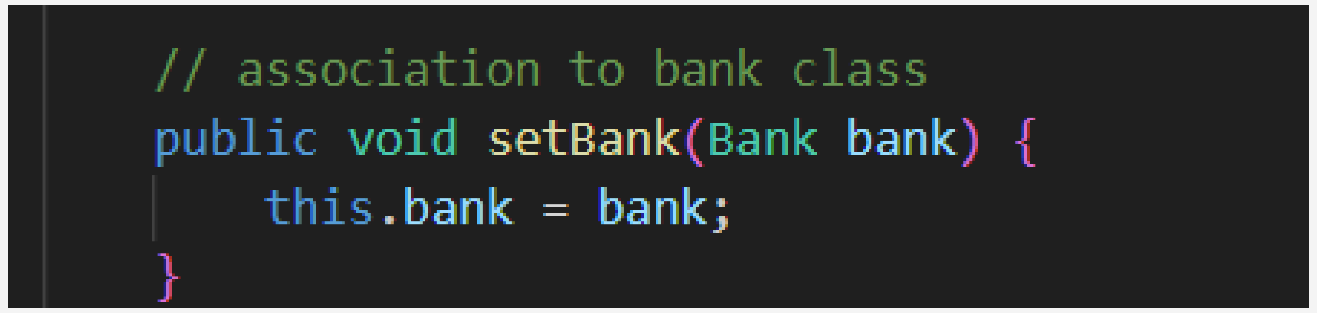
IN CODE



```
public void setUser(Users user) {
    this.user = user;
}
```



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



```
// association to bank class
public void setBank(Bank bank) {
    this.bank = bank;
}
```

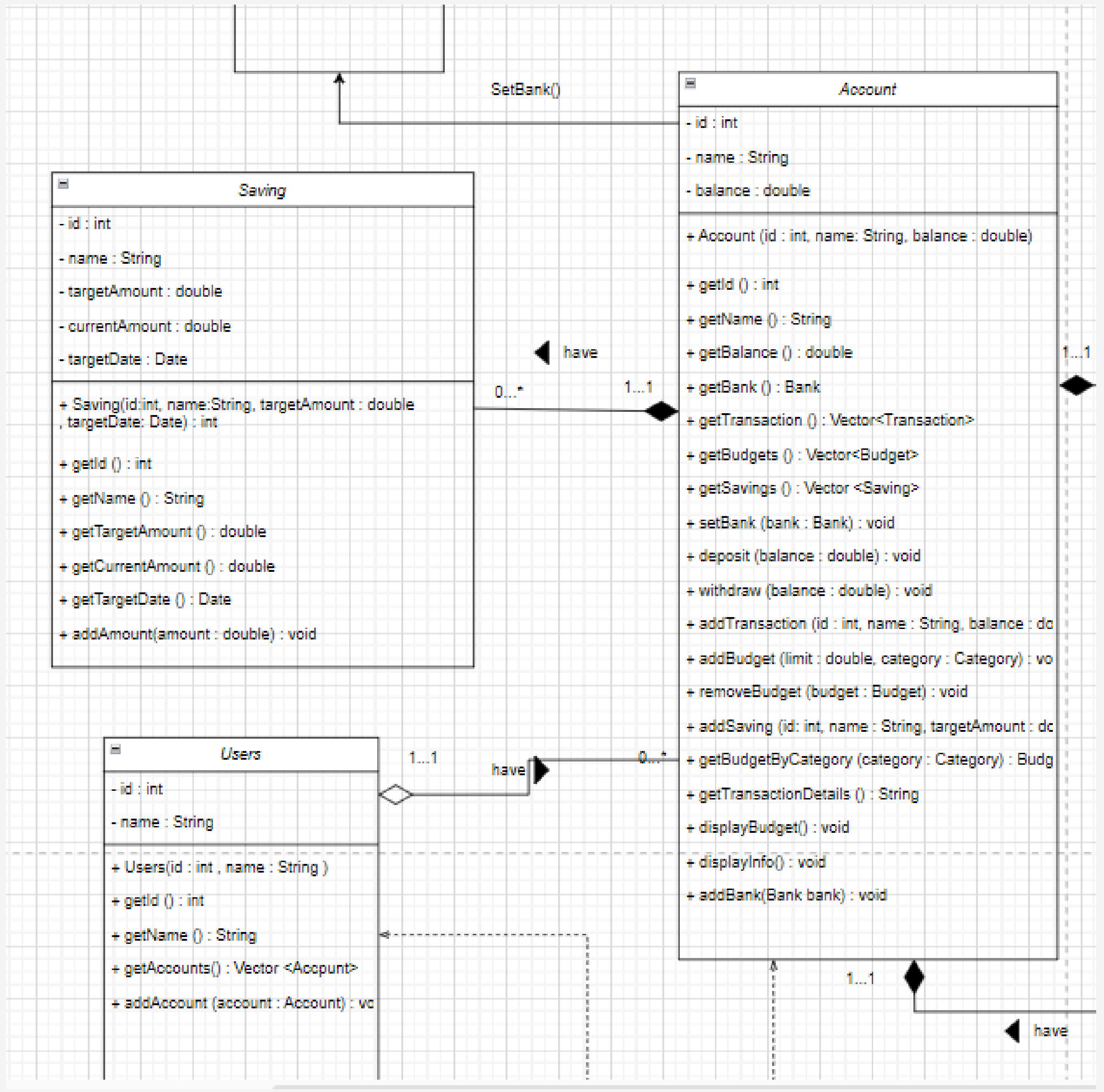
Association require and additional method inside the class for another object to be “associated” with the class via method calling and passing argument object

NAME OF PROJECT

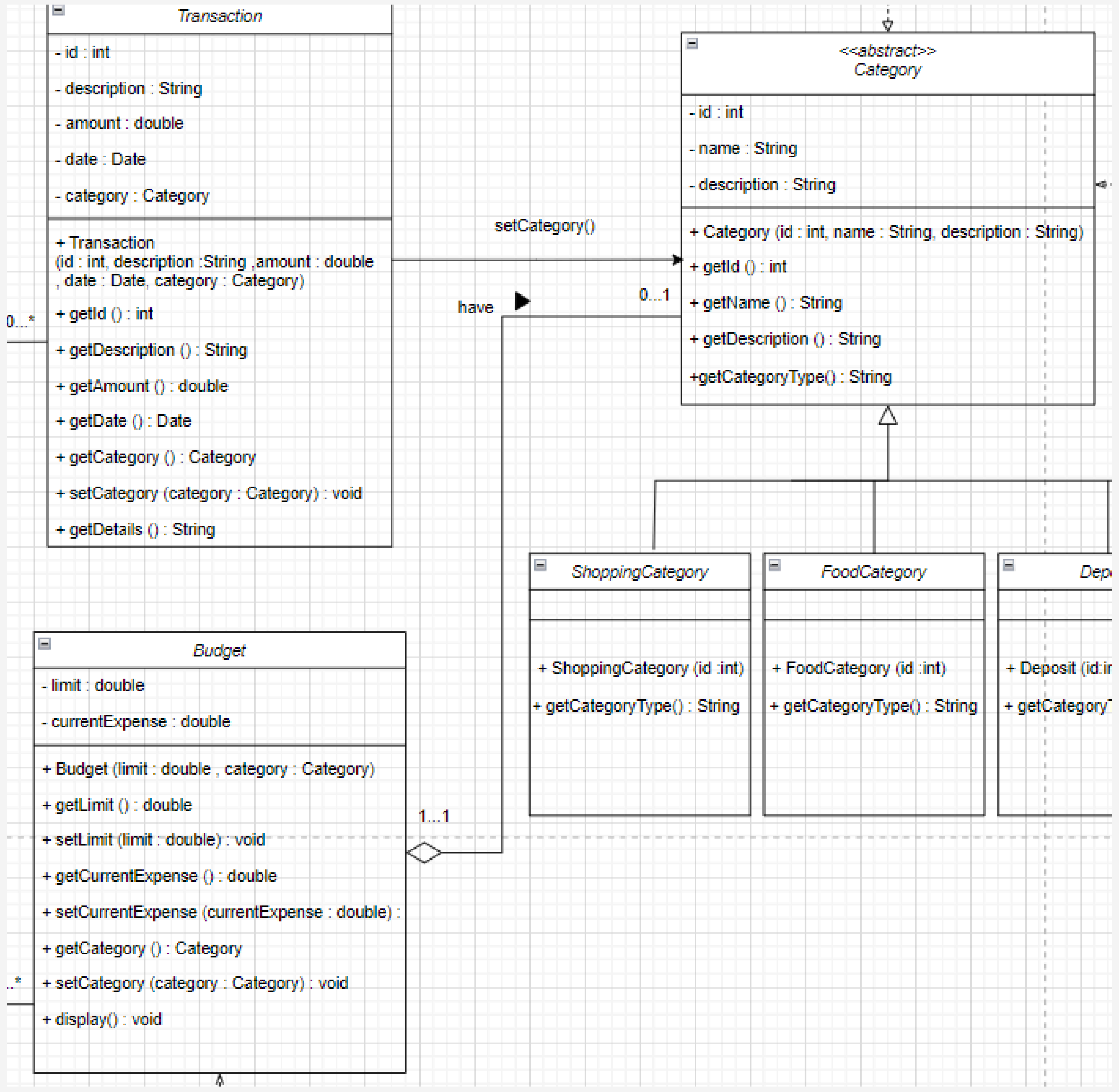
Finance Manager

AGGREGATION

- Aggregation between class Budget and Category
- Aggregation between class Users and Account



SetBank()
Account
- id : int
- name : String
- balance : double
+ Account (id : int, name: String, balance : double)
+ getId () : int
+ getName () : String
+ getBalance () : double
+ getBank () : Bank
+ getTransaction () : Vector<Transaction>
+ getBudgets () : Vector<Budget>
+ getSavings () : Vector <Saving>
+ setBank (bank : Bank) : void
+ deposit (balance : double) : void
+ withdraw (balance : double) : void
+ addTransaction (id : int, name : String, balance : do
+ addBudget (limit : double, category : Category) : vo
+ removeBudget (budget : Budget) : void
+ addSaving (id: int, name : String, targetAmount : dc
+ getBudgetByCategory (category : Category) : Budg
+ getTransactionDetails () : String
+ displayBudget() : void
+ displayInfo() : void
+ addBank(Bank bank) : void
Saving
- id : int
- name : String
- targetAmount : double
- currentAmount : double
- targetDate : Date
+ Saving(id:int, name:String, targetAmount : double , targetDate: Date) : int
+ getId () : int
+ getName () : String
+ getTargetAmount () : double
+ getCurrentAmount () : double
+ getTargetDate () : Date
+ addAmount(amount : double) : void
have
0...*
1...1
1...1
Users
- id : int
- name : String
+ Users(id : int , name : String)
+ getId () : int
+ getName () : String
+ getAccounts() : Vector <Account>
+ addAccount (account : Account) : vo
1...1
have
0...*
1...1
have

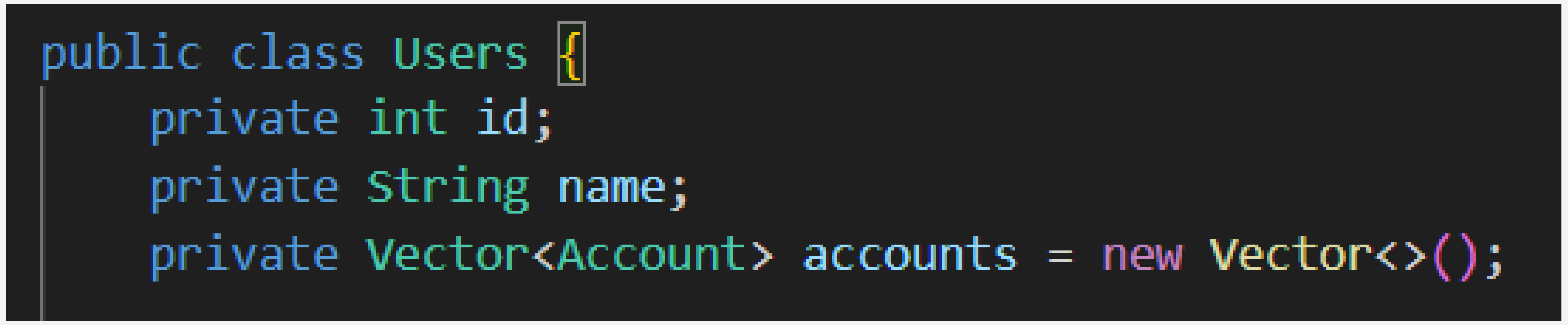


Transaction
- id : int
- description : String
- amount : double
- date : Date
- category : Category
+ Transaction (id : int, description :String ,amount : double , date : Date, category : Category)
+ getId () : int
+ getDescription () : String
+ getAmount () : double
+ getDate () : Date
+ getCategory () : Category
+ setCategory (category : Category) : void
+ getDetails () : String
setCategory()
have
0...1
0...*
<<abstract>> Category
- id : int
- name : String
- description : String
+ Category (id : int, name : String, description : String)
+ getId () : int
+ getName () : String
+ getDescription () : String
+getCategoryType() : String
ShoppingCategory
+ ShoppingCategory (id :int)
+ getCategoryType() : String
FoodCategory
+ FoodCategory (id :int)
+ getCategoryType() : String
Dep
+ Deposit (id:i
+ getCategory
Budget
- limit : double
- currentExpense : double
+ Budget (limit : double , category : Category)
+ getLimit () : double
+ setLimit (limit : double) : void
+ getCurrentExpense () : double
+ setCurrentExpense (currentExpense : double) :
+ getCategory () : Category
+ setCategory (category : Category) : void
+ display() : void
1...1
..*

NAME OF PROJECT
Finance Manager

AGGREGATION

IN CODE

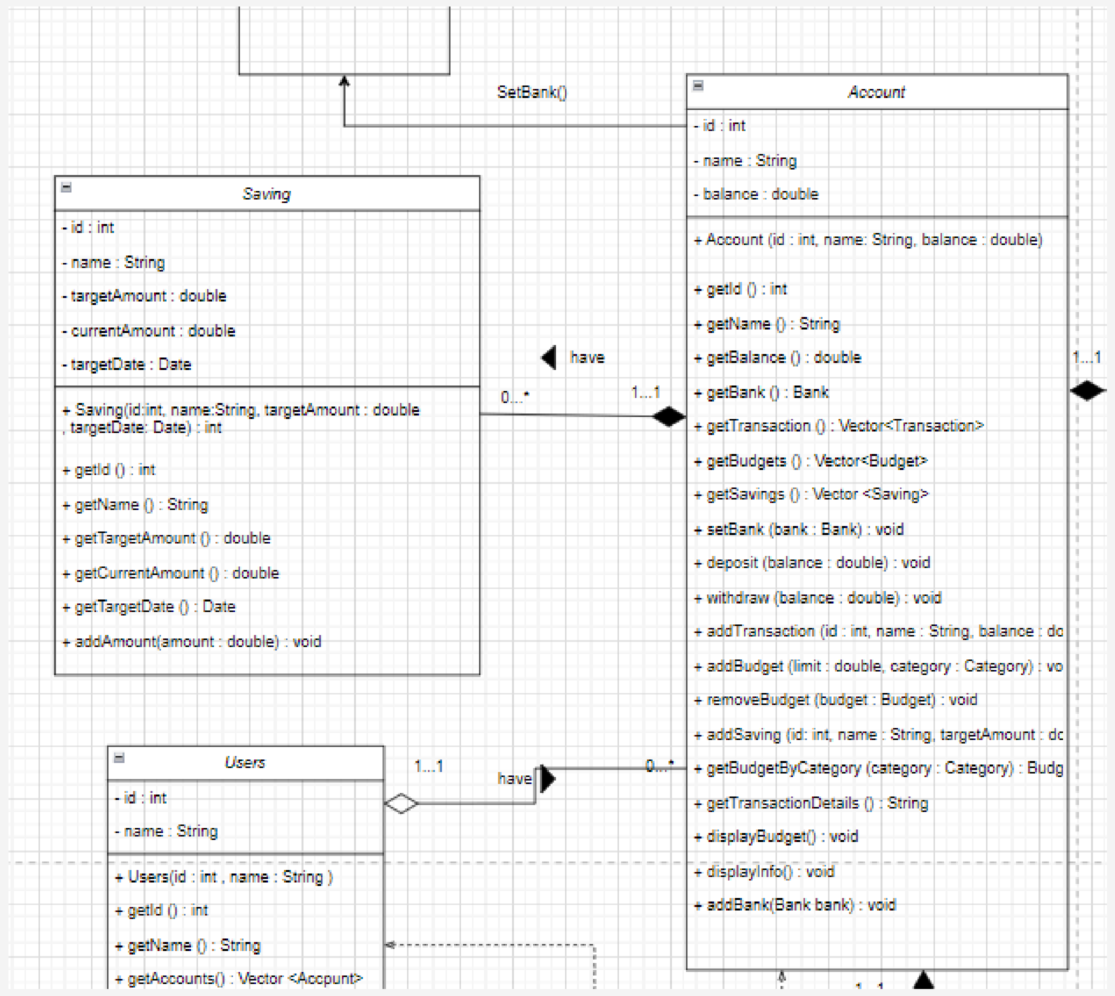

```
public class Users {
    private int id;
    private String name;
    private Vector<Account> accounts = new Vector<>();
```

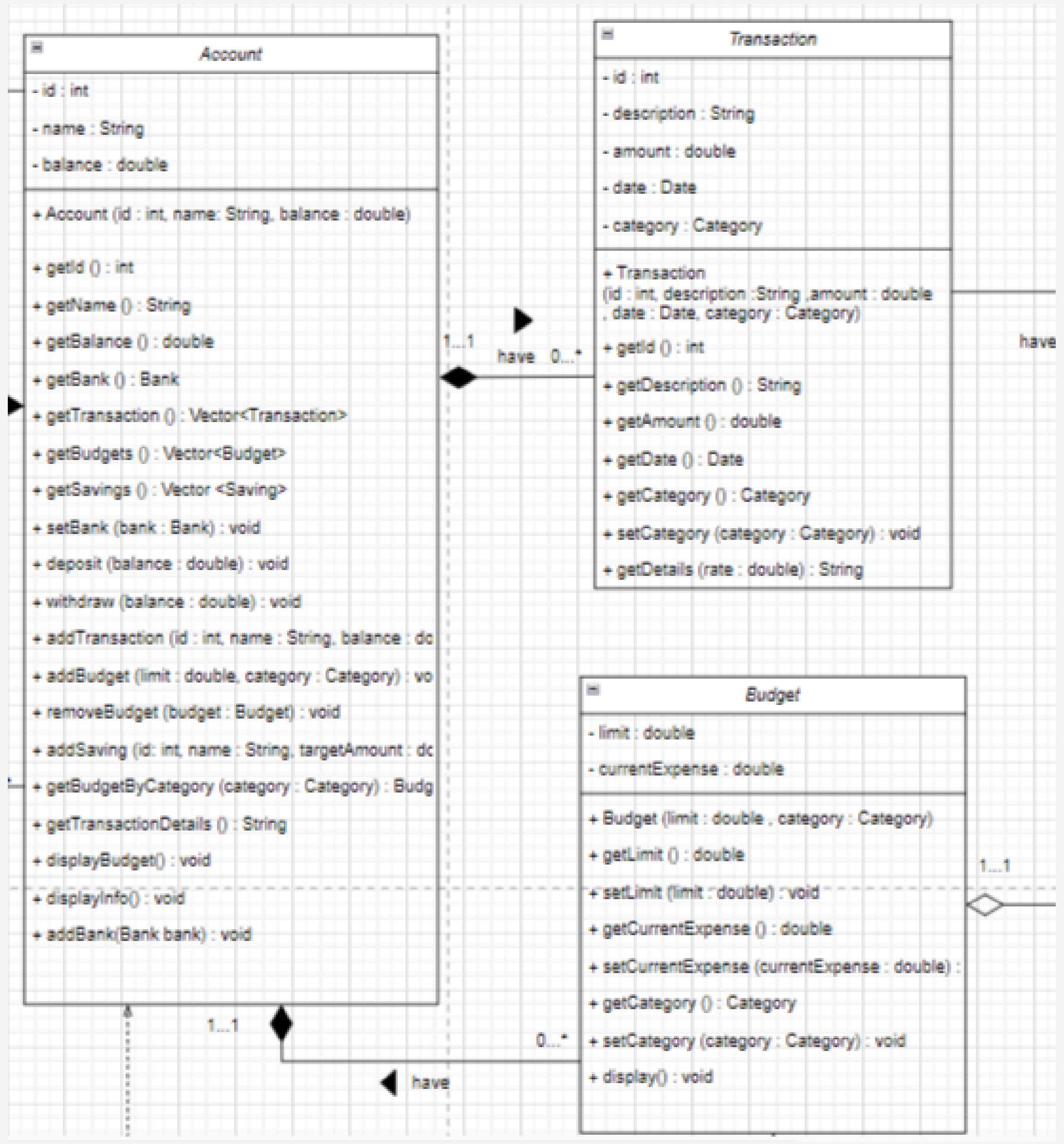
Aggregation are when an object of another class are initialized when it is called for this example when object users are created,object Accounts would automatically created.

NAME OF PROJECT
Finance Manager

COMPOSITION

- Composition between class Saving and Account
- Composition between class Account and Transaction
- Composition between class Account and Budget

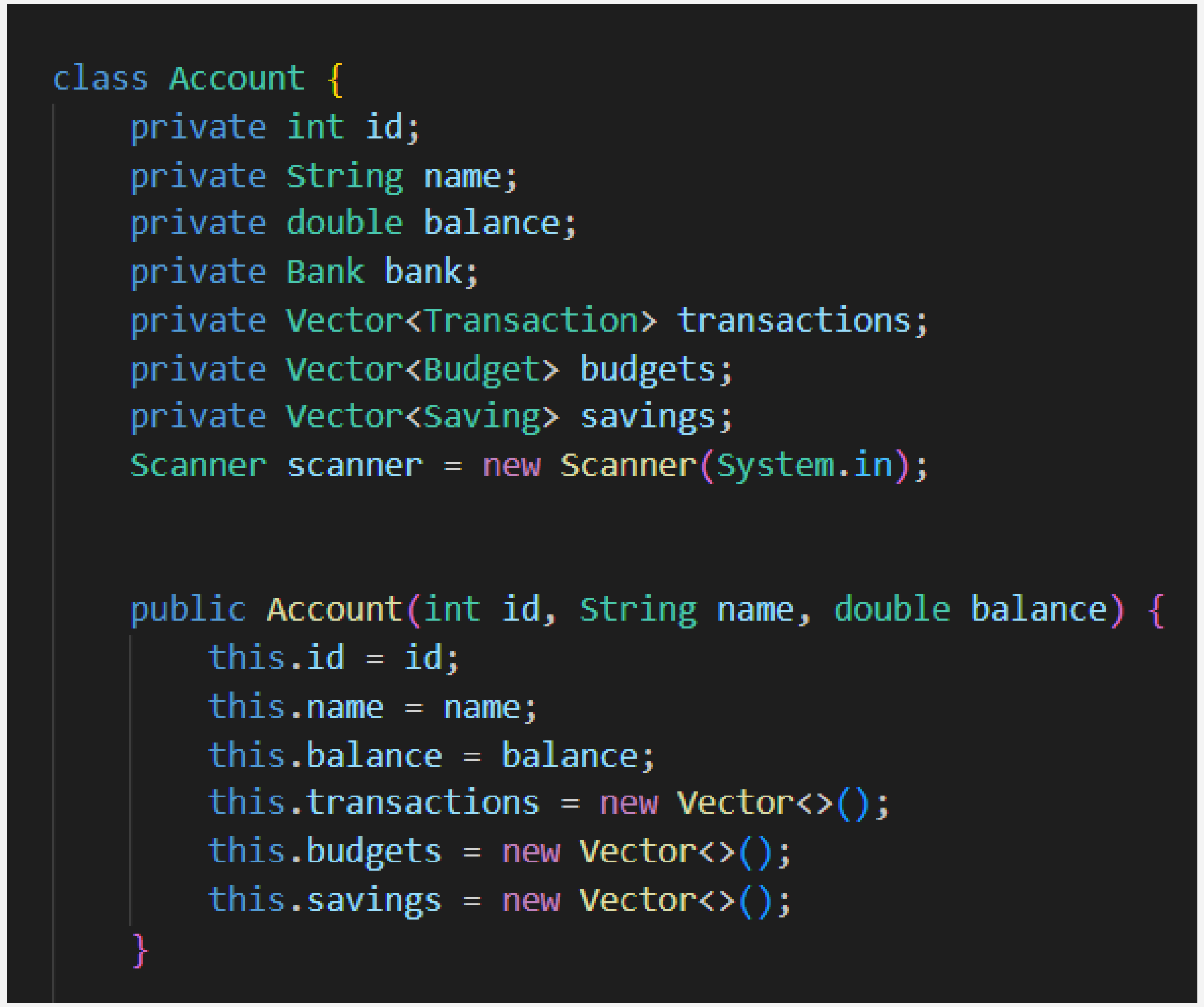

SetBank()
Account
- id : int
- name : String
- balance : double
+ Account (id : int, name: String, balance : double)
+ getId () : int
+ getName () : String
+ getBalance () : double
+ getBank () : Bank
+ getTransaction () : Vector<Transaction>
+ getBudgets () : Vector<Budget>
+ getSavings () : Vector <Saving>
+ setBank (bank : Bank) : void
+ deposit (balance : double) : void
+ withdraw (balance : double) : void
+ addTransaction (id : int, name : String, balance : dc
+ addBudget (limit : double, category : Category) : vo
+ removeBudget (budget : Budget) : void
+ addSaving (id: int, name : String, targetAmount : dc
+ getBudgetByCategory (category : Category) : Budg
+ getTransactionDetails () : String
+ displayBudget() : void
+ displayInfo() : void
+ addBank(Bank bank) : void
Saving
- id : int
- name : String
- targetAmount : double
- currentAmount : double
- targetDate : Date
+ Saving(id:int, name:String, targetAmount : double , targetDate: Date) : int
+ getId () : int
+ getName () : String
+ getTargetAmount () : double
+ getCurrentAmount () : double
+ getTargetDate () : Date
+ addAmount(amount : double) : void
have
0...*
1...1
1...1
Users
- id : int
- name : String
+ Users(id : int , name : String)
+ getId () : int
+ getName () : String
+ getAccounts() : Vector <Accpunt>
1...1
have
0...*


Account
- id : int
- name : String
- balance : double
+ Account (id : int, name: String, balance : double)
+ getId () : int
+ getName () : String
+ getBalance () : double
+ getBank () : Bank
+ getTransaction () : Vector<Transaction>
+ getBudgets () : Vector<Budget>
+ getSavings () : Vector <Saving>
+ setBank (bank : Bank) : void
+ deposit (balance : double) : void
+ withdraw (balance : double) : void
+ addTransaction (id : int, name : String, balance : dc
+ addBudget (limit : double, category : Category) : vo
+ removeBudget (budget : Budget) : void
+ addSaving (id: int, name : String, targetAmount : dc
+ getBudgetByCategory (category : Category) : Budg
+ getTransactionDetails () : String
+ displayBudget() : void
+ displayInfo() : void
+ addBank(Bank bank) : void
Transaction
- id : int
- description : String
- amount : double
- date : Date
- category : Category
+ Transaction (id : int, description :String ,amount : double , date : Date, category : Category)
+ getId () : int
+ getDescription () : String
+ getAmount () : double
+ getDate () : Date
+ getCategory () : Category
+ setCategory (category : Category) : void
+ getDetails (rate : double) : String
1...1
have
0...*
have
Budget
- limit : double
- currentExpense : double
+ Budget (limit : double , category : Category)
+ getLimit () : double
+ setLimit (limit : double) : void
+ getCurrentExpense () : double
+ setCurrentExpense (currentExpense : double) :
+ getCategory () : Category
+ setCategory (category : Category) : void
+ display() : void
1...1
1...1
0...*
have

NAME OF PROJECT
Finance Manager

COMPOSITION

IN CODE



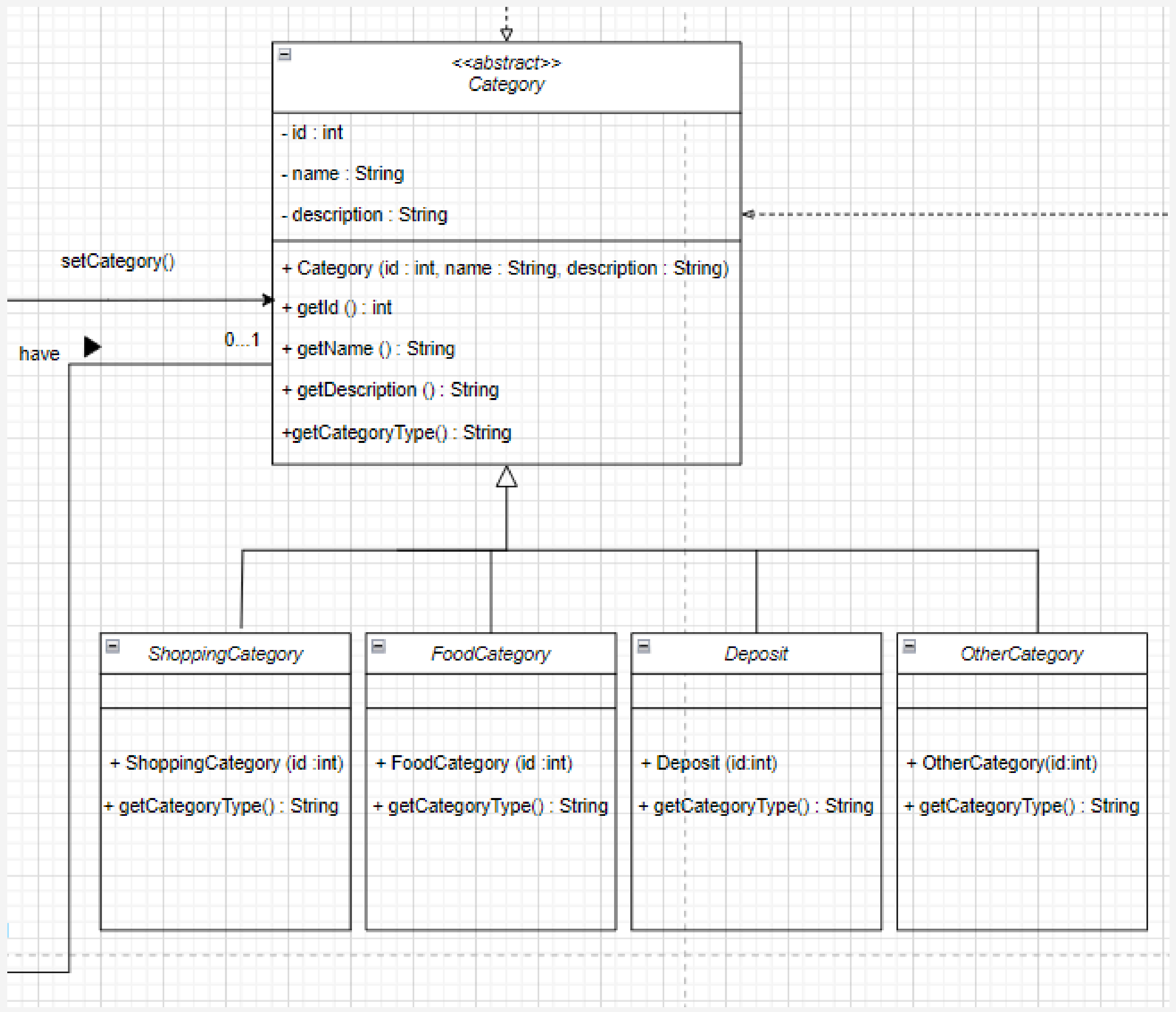
```
class Account {
    private int id;
    private String name;
    private double balance;
    private Bank bank;
    private Vector<Transaction> transactions;
    private Vector<Budget> budgets;
    private Vector<Saving> savings;
    Scanner scanner = new Scanner(System.in);


    public Account(int id, String name, double balance) {
        this.id = id;
        this.name = name;
        this.balance = balance;
        this.transactions = new Vector<>();
        this.budgets = new Vector<>();
        this.savings = new Vector<>();
    }
```

Compositions are when the class are initialized in the constructor, it represent strong relationship between classes and one cannot exist without another in this example saving, budget and transaction cannot exist without an account.

NAME OF PROJECT
Finance Manager

INHERITANCE

- Inheritance of Class ShoppingCategory,FoodCategory,Deposit,OtherCategory from class Category

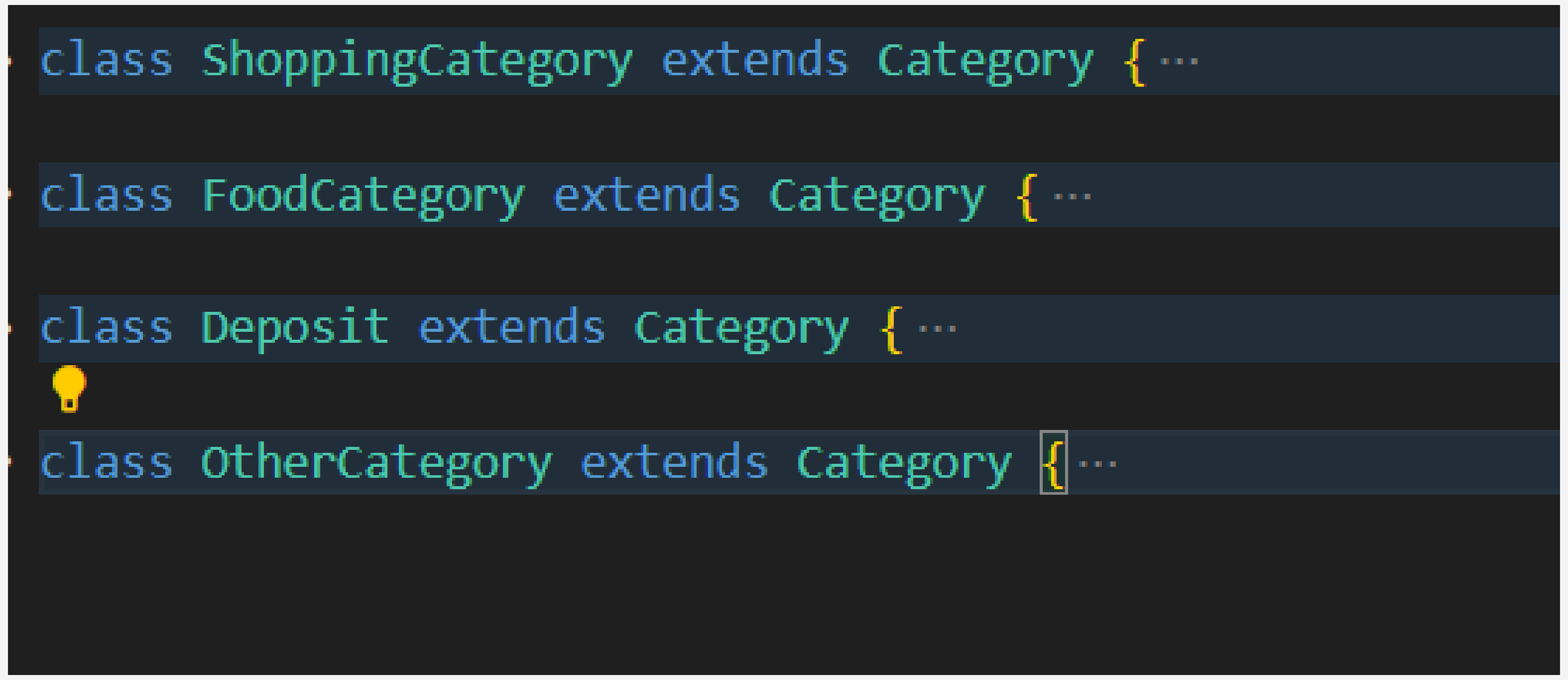


<<abstract>>
Category
- id : int
- name : String
- description : String
+ Category (id : int, name : String, description : String)
+ getId () : int
+ getName () : String
+ getDescription () : String
+getCategoryType() : String
setCategory()
have
0...1
ShoppingCategory
+ ShoppingCategory (id :int)
+ getCategoryType() : String
FoodCategory
+ FoodCategory (id :int)
+ getCategoryType() : String
Deposit
+ Deposit (id:int)
+ getCategoryType() : String
OtherCategory
+ OtherCategory(id:int)
+ getCategoryType() : String

NAME OF PROJECT
Finance Manager

INHERITANCE

IN CODE



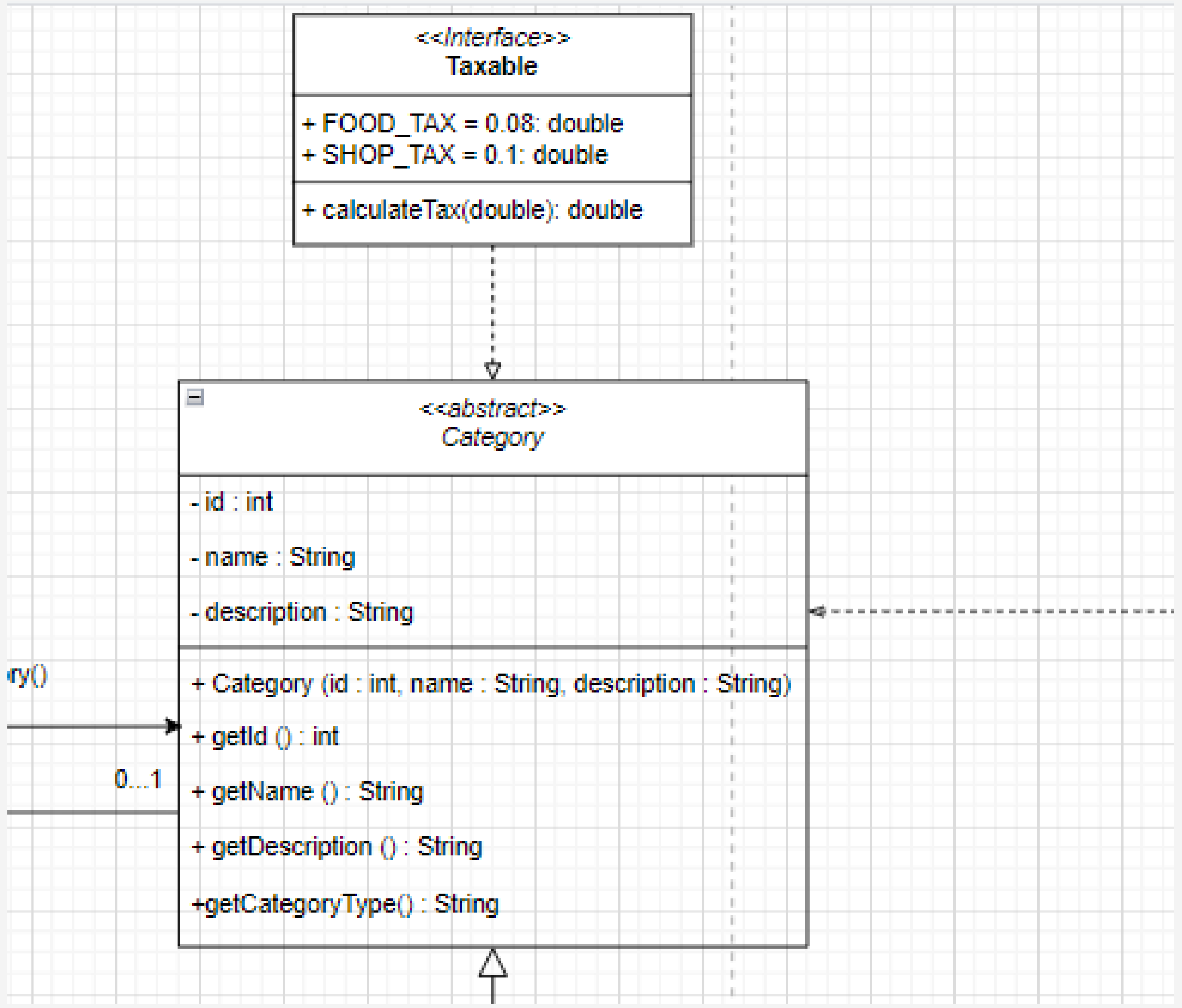
```
class ShoppingCategory extends Category {···

class FoodCategory extends Category {···

class Deposit extends Category {···

class OtherCategory extends Category {···
```

Inheritance are when the classes inherit (can use) the method and variable of the class that they are subclass of, in this code all of these are subclass and the parent class is Category.

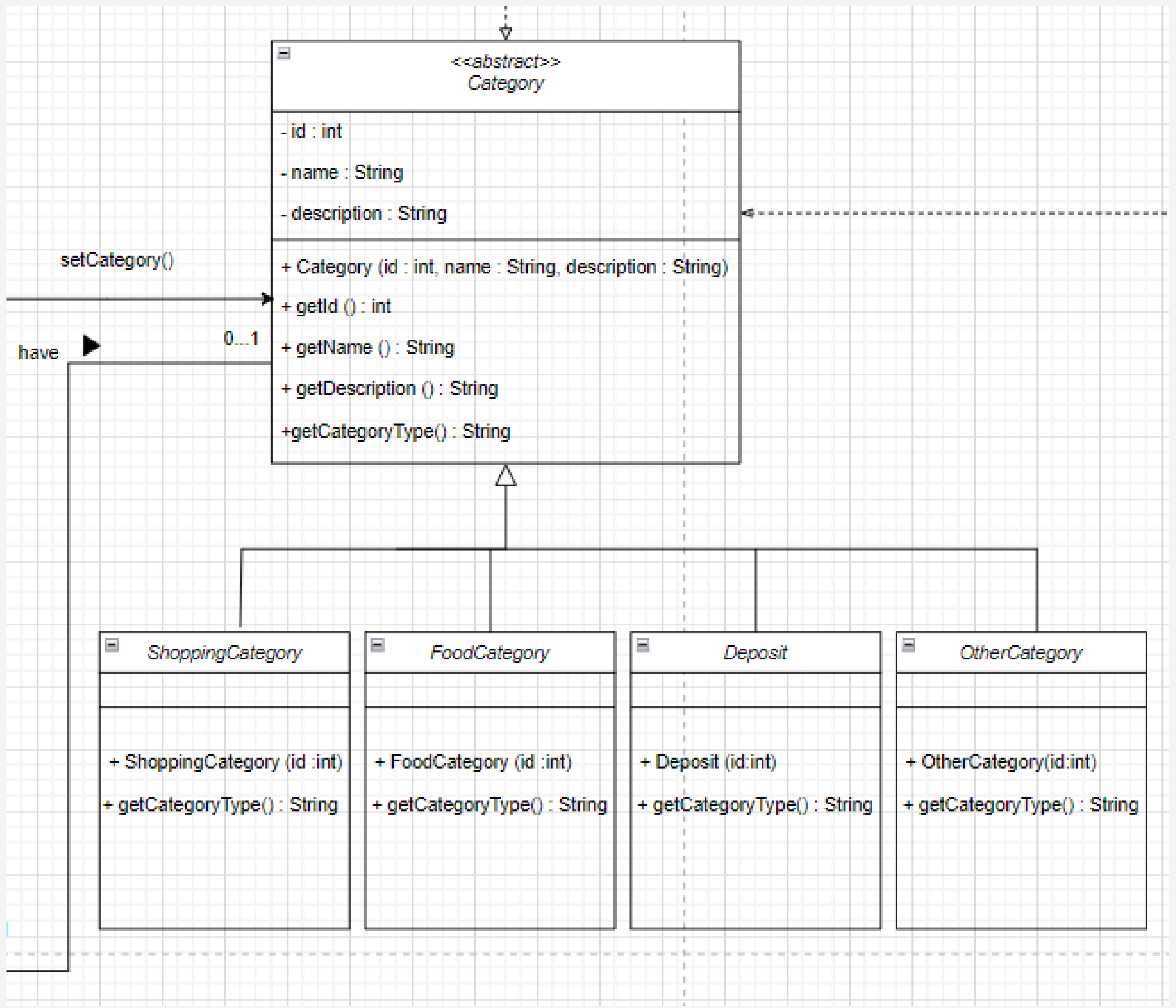
NAME OF PROJECT
Finance Manager

POLYMORPHISM

- **Polymorphism of Interface Taxable implemented on Category**
- **Polymorphism of Abstract class Category**



<<Interface>>
Taxable
+ FOOD_TAX = 0.08: double
+ SHOP_TAX = 0.1: double
+ calculateTax(double): double
<<abstract>>
Category
- id : int
- name : String
- description : String
+ Category (id : int, name : String, description : String)
+ getId () : int
+ getName () : String
+ getDescription () : String
+getCategoryType() : String
ry()
0...1

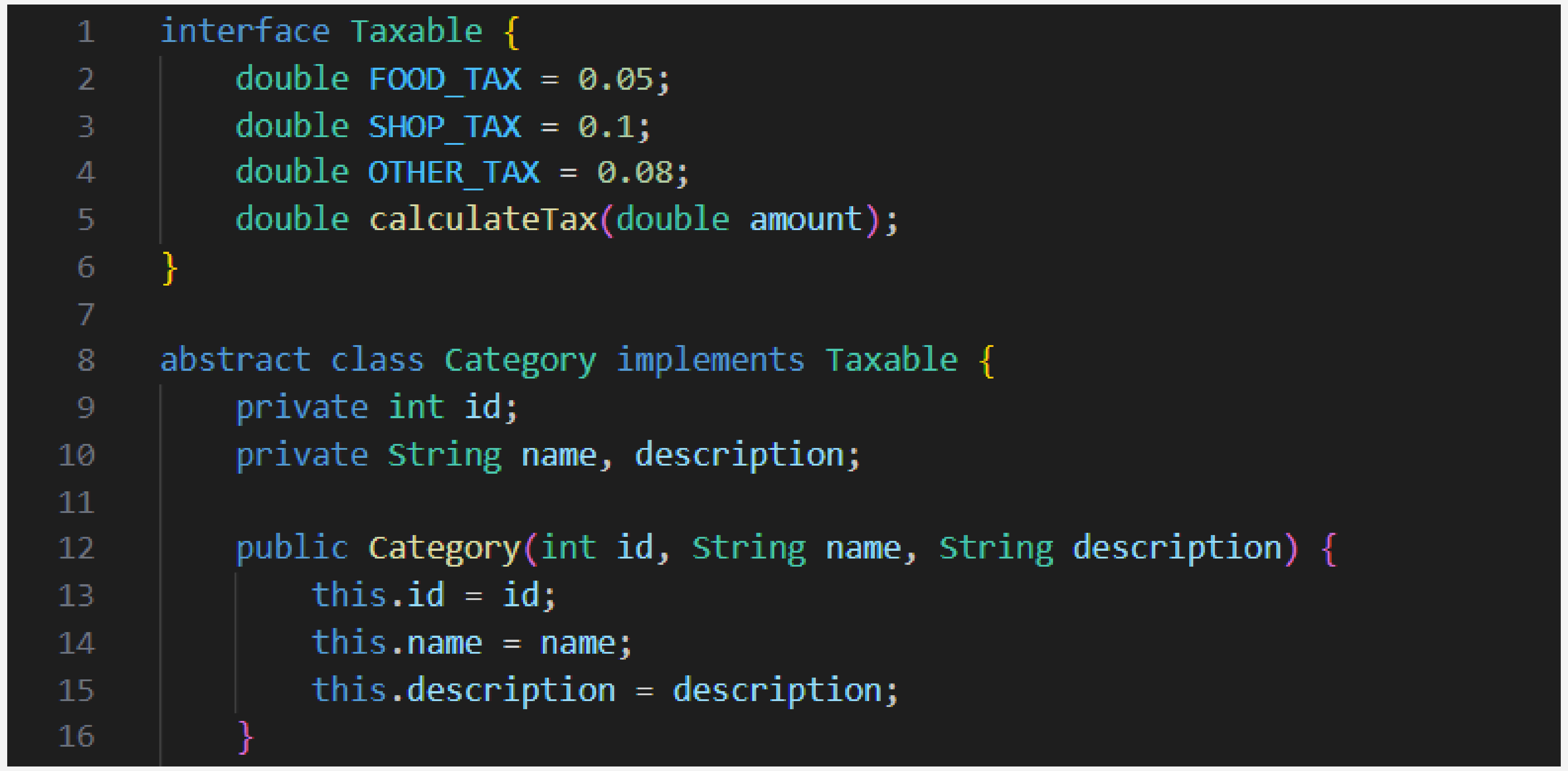


<<abstract>>
Category
- id : int
- name : String
- description : String
+ Category (id : int, name : String, description : String)
+ getId () : int
+ getName () : String
+ getDescription () : String
+getCategoryType() : String
setCategory()
have
0...1
ShoppingCategory
+ ShoppingCategory (id :int)
+ getCategoryType() : String
FoodCategory
+ FoodCategory (id :int)
+ getCategoryType() : String
Deposit
+ Deposit (id:int)
+ getCategoryType() : String
OtherCategory
+ OtherCategory(id:int)
+ getCategoryType() : String

NAME OF PROJECT
Finance Manager

POLYMORPHISM

IN CODE



```
interface Taxable {
    double FOOD_TAX = 0.05;
    double SHOP_TAX = 0.1;
    double OTHER_TAX = 0.08;
    double calculateTax(double amount);
}

abstract class Category implements Taxable {
    private int id;
    private String name, description;

    public Category(int id, String name, String description) {
        this.id = id;
        this.name = name;
        this.description = description;
    }
```

Interface of Taxable which only contain constant variable and abstract method class,this code class category implement the Taxable interface, this require for class Category to have all the abstract method of interface Varible.

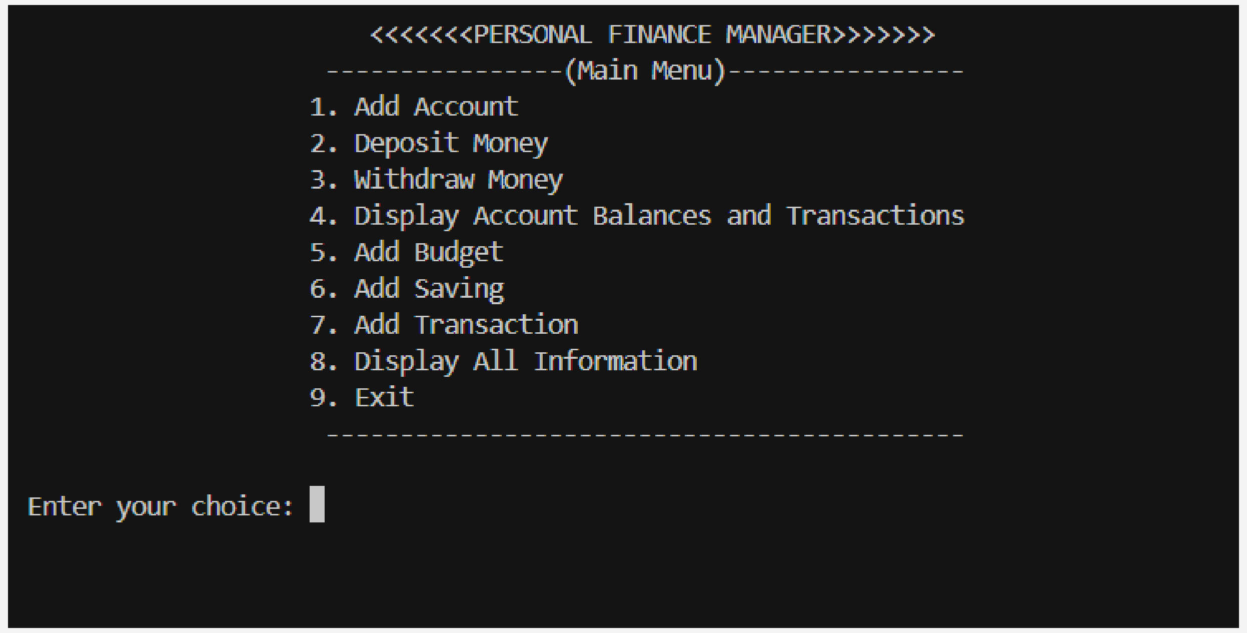
DEMONSTRATION

DATE
26/06/2024

NAME OF PROJECT
Finance Manager

FEATURES

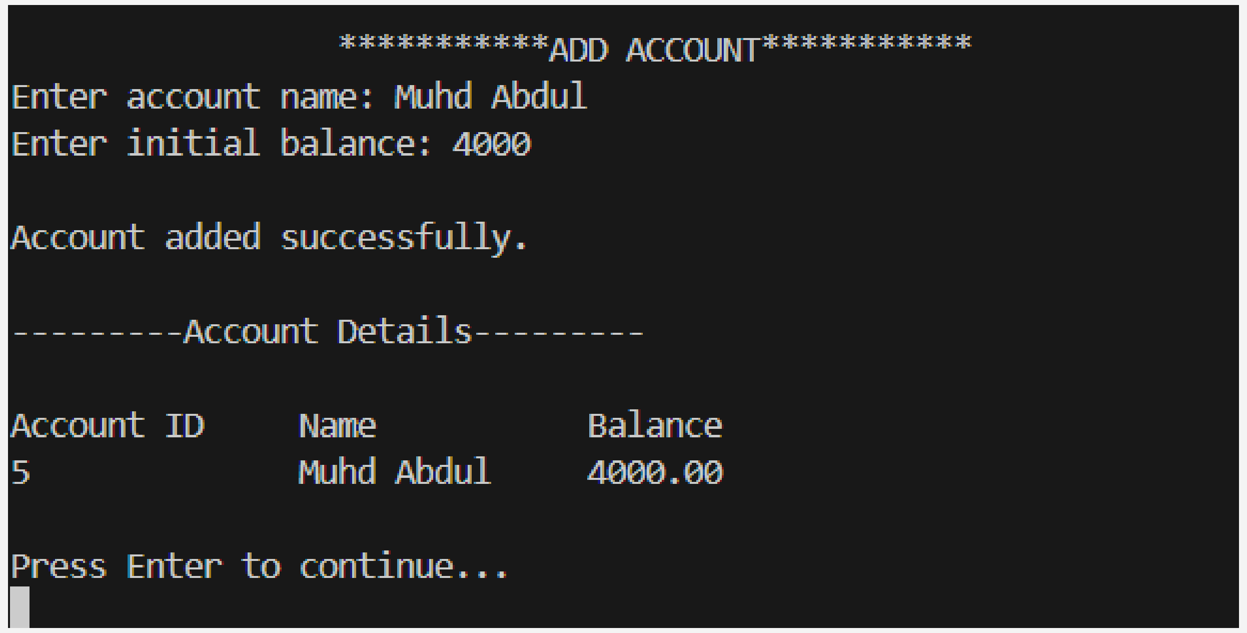
01. MAIN MENU



```
            <<<<<<<PERSONAL FINANCE MANAGER>>>>>>>
           ----------------(Main Menu)----------------
          1. Add Account
          2. Deposit Money
          3. Withdraw Money
          4. Display Account Balances and Transactions
          5. Add Budget
          6. Add Saving
          7. Add Transaction
          8. Display All Information
          9. Exit
           -------------------------------------------

Enter your choice:
```

Main menu for user to choose which operation they want to select.

02. ADD ACCOUNT



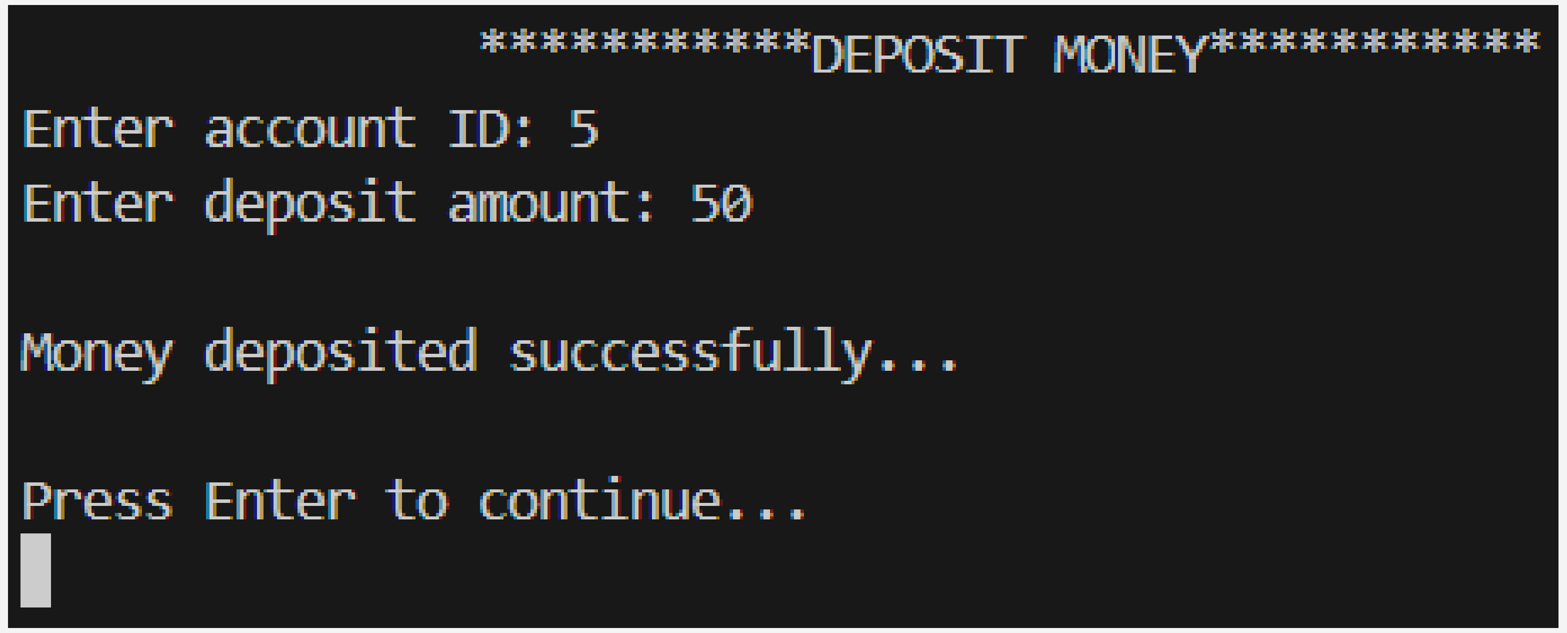
```
                ***********ADD ACCOUNT***********
Enter account name: Muhd Abdul
Enter initial balance: 4000

Account added successfully.

---------Account Details---------

Account ID     Name           Balance
5              Muhd Abdul     4000.00

Press Enter to continue...
```

For this option, user need to fill the information regarding the new account.

PRESENTED BY
CodeHub

CONTINUING...

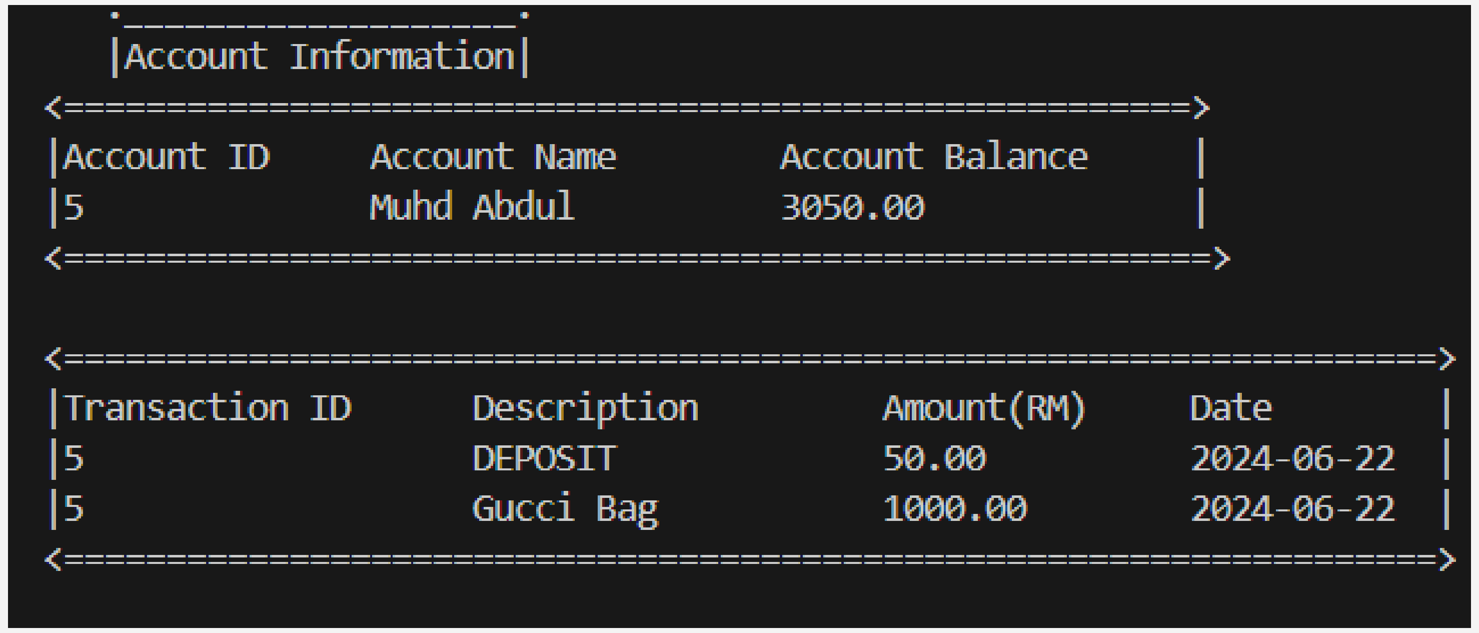
03. DEPOSIT MONEY



```
               ***********DEPOSIT MONEY***********
Enter account ID: 5
Enter deposit amount: 50

Money deposited successfully...

Press Enter to continue...
```

For the deposit option, user need to fill the information regarding the deposit money.

04. DISPLAY ACCOUNT AND TRANSACTION

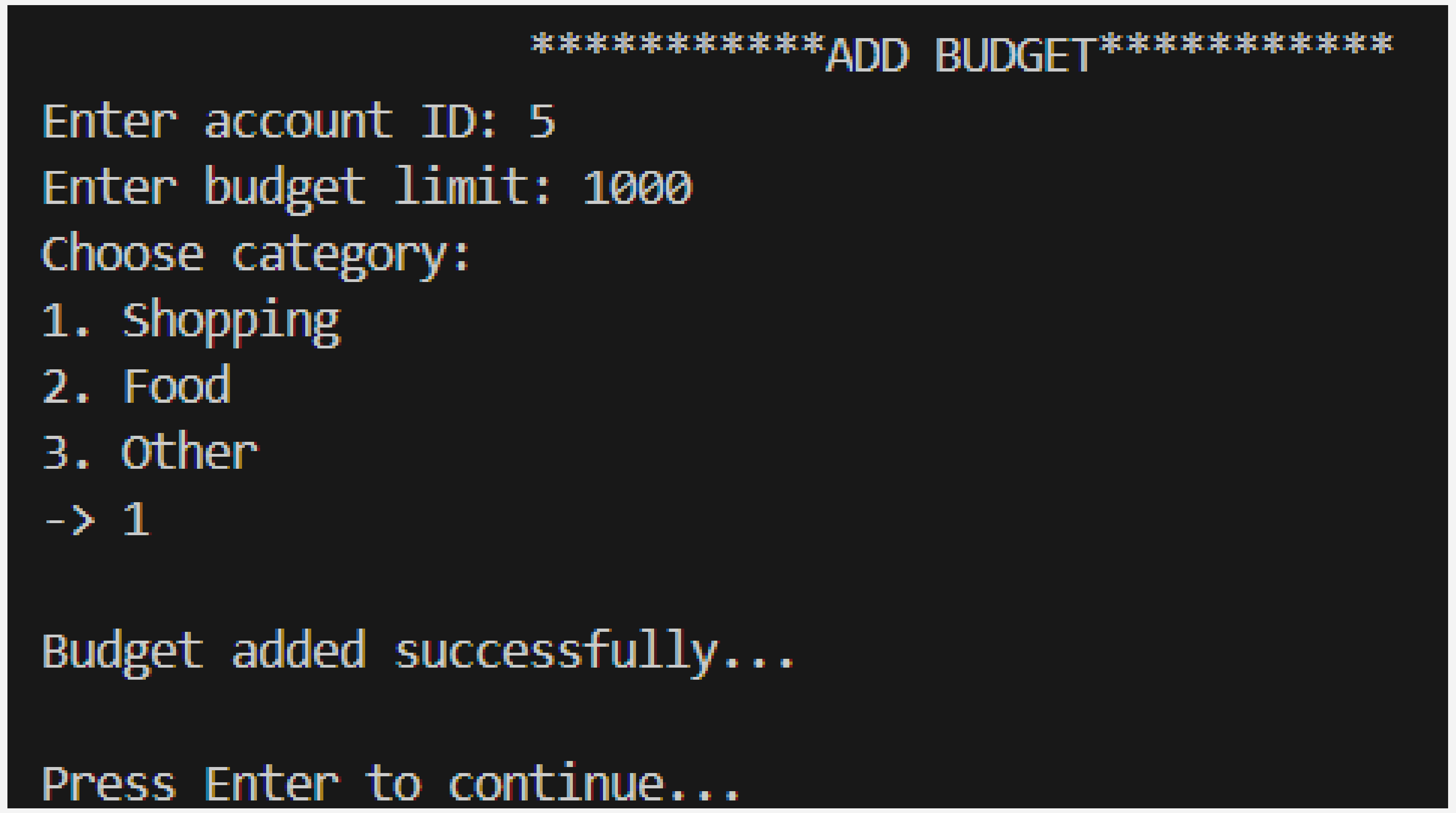


```
   .___________________.
   |Account Information|
<=======================================================>
|Account ID     Account Name        Account Balance     |
|5              Muhd Abdul          3050.00             |
<========================================================>


<==================================================================>
|Transaction ID      Description         Amount(RM)     Date        |
|5                   DEPOSIT             50.00          2024-06-22  |
|5                   Gucci Bag           1000.00        2024-06-22  |
<==================================================================>
```

For the display account and transaction option, the information about account and their respective transaction will be displayed.

CONTINUING...

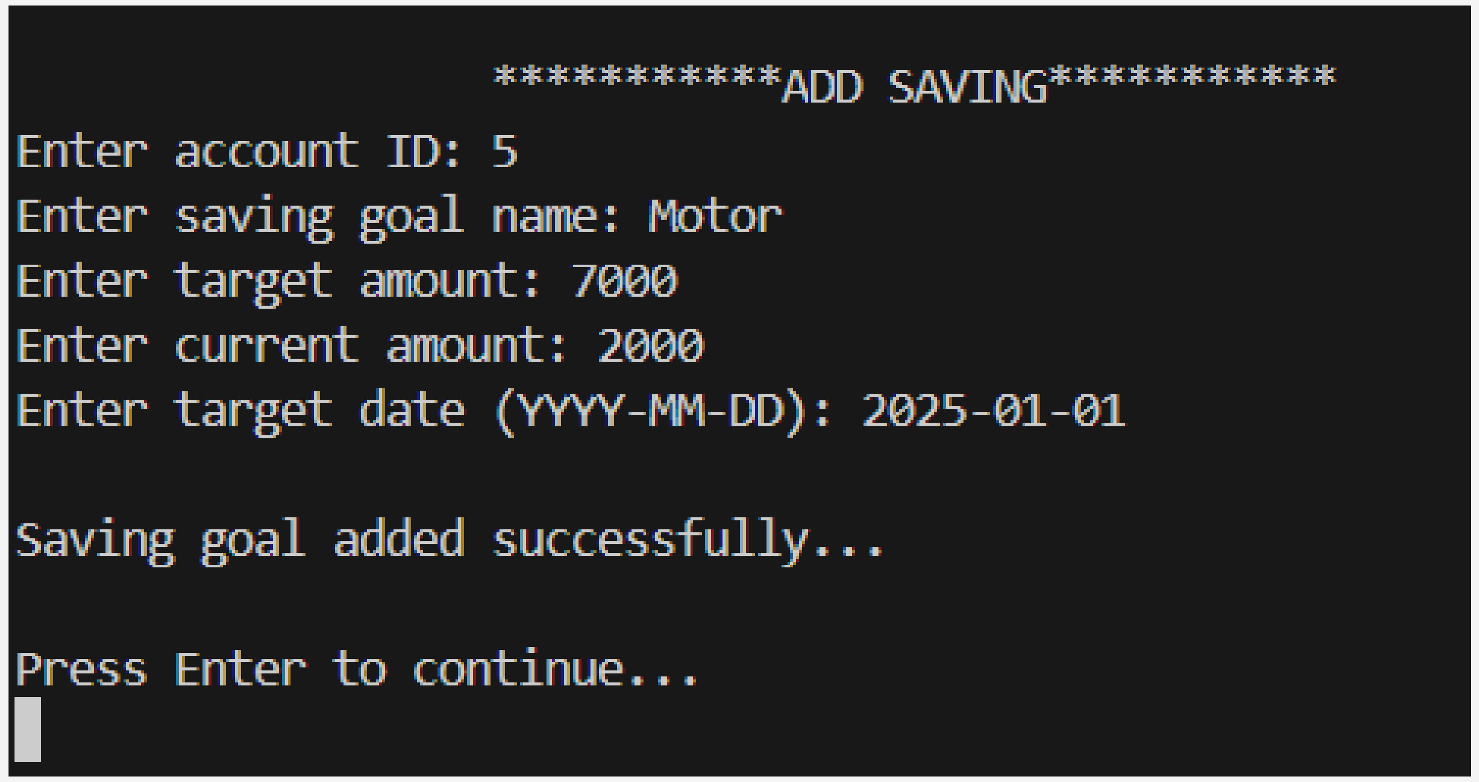
06. ADD BUDGET



```
                    ***********ADD BUDGET***********
Enter account ID: 5
Enter budget limit: 1000
Choose category:
1. Shopping
2. Food
3. Other
-> 1

Budget added successfully...

Press Enter to continue...
```

For this option, user need to make a specific budget limit with category type.

07. ADD SAVING

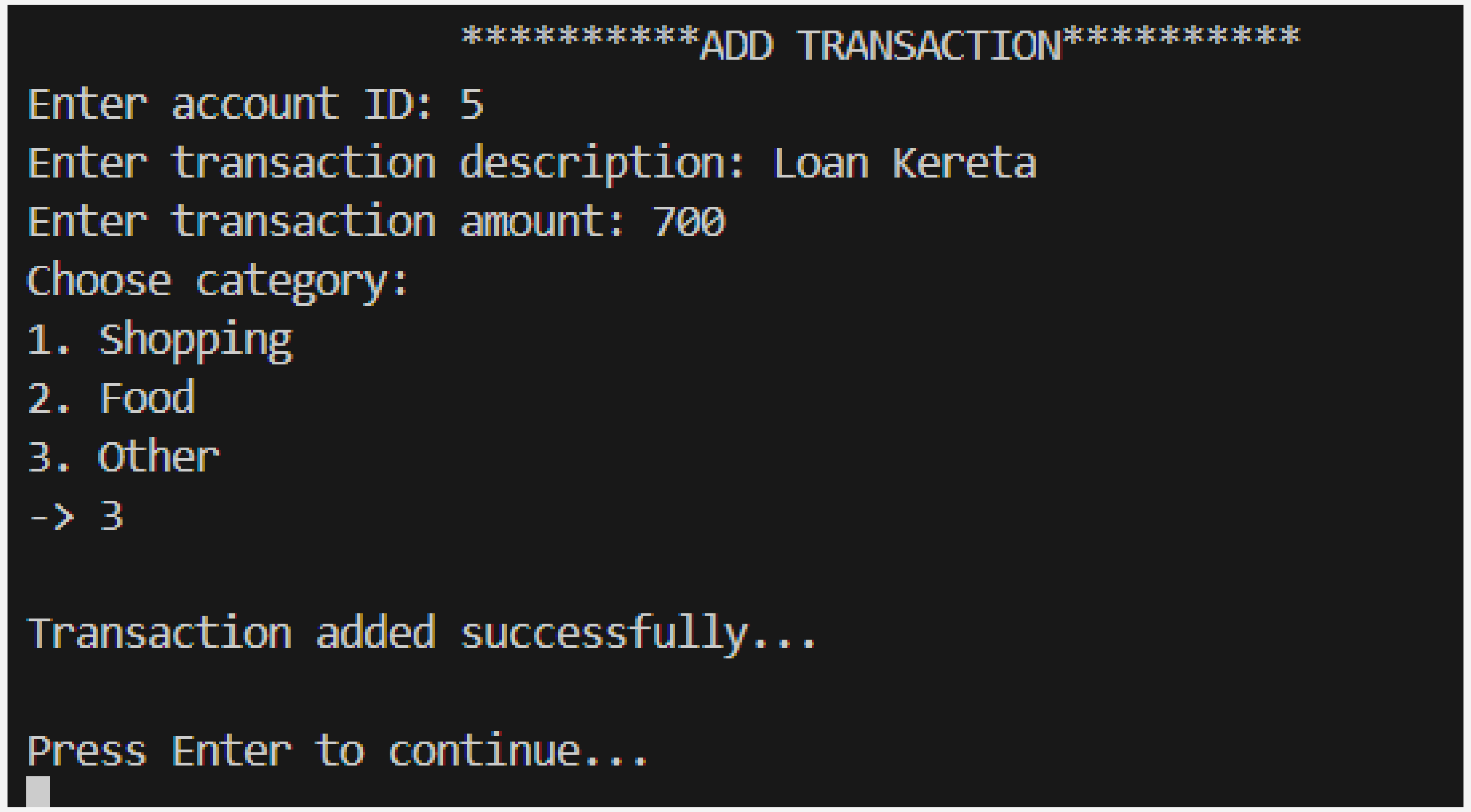


```
                    ***********ADD SAVING***********
Enter account ID: 5
Enter saving goal name: Motor
Enter target amount: 7000
Enter current amount: 2000
Enter target date (YYYY-MM-DD): 2025-01-01

Saving goal added successfully...

Press Enter to continue...
```

For this option, user need to make a specific saving goal with other information about the saving.

CONTINUING...

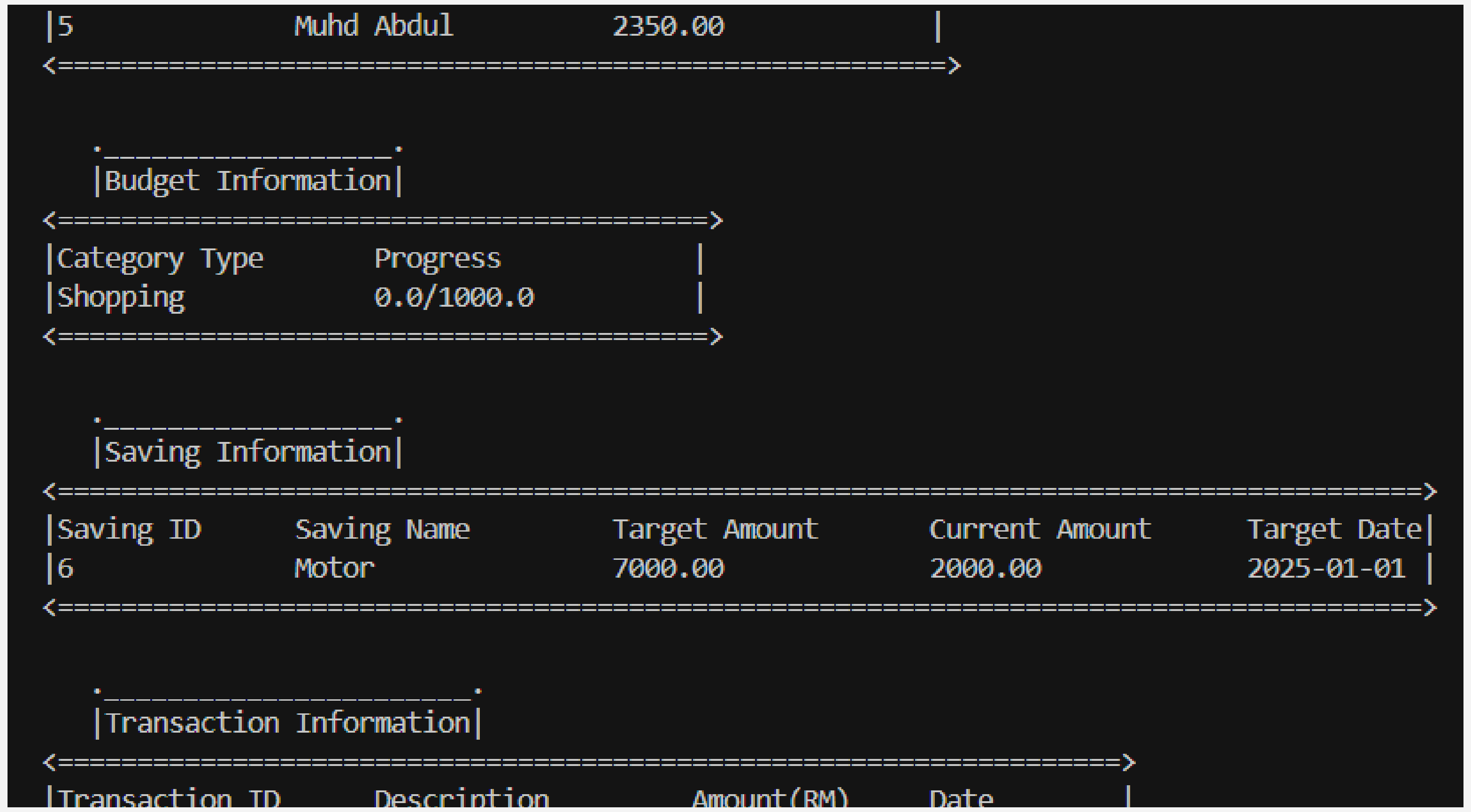
08. ADD TRANSACTION



```
                    **********ADD TRANSACTION**********
Enter account ID: 5
Enter transaction description: Loan Kereta
Enter transaction amount: 700
Choose category:
1. Shopping
2. Food
3. Other
-> 3

Transaction added successfully...

Press Enter to continue...
```

For adding transaction, the required information regarding transaction is needed from the user

09. DISPLAY ALL INFORMATION



```
|5              Muhd Abdul          2350.00              |
<========================================================>


   ._________________.
   |Budget Information|
<=========================================>
|Category Type       Progress            |
|Shopping            0.0/1000.0          |
<=========================================>


   ._________________.
   |Saving Information|
<======================================================================================>
|Saving ID      Saving Name         Target Amount       Current Amount      Target Date|
|6              Motor               7000.00             2000.00             2025-01-01 |
<======================================================================================>


   ._______________________.
   |Transaction Information|
<===================================================================>
|Transaction ID      Description         Amount(RM)     Date        |
```

Display all information regarding user accounts which is balance, budget, saving, and transaction.

DATE
26/6/2024

THANK YOU

NAME OF GROUP
CodeHub