Personal Budget Manager: Project Description

C++ Project: Submit on CTL

Due Date: 2023/12/14

Project Overview

In this project, students will develop a **Personal Budget Manager** application using. This project is designed to introduce fundamental programming concepts, object-oriented programming principles, and provides a practical application to manage personal finances.

- The UML is just a sample
- Creativity is encouraged
- DO NOT COPY
  - If you plan to work in a team, send me a notice on CTL
  - Max number is 3
  - All must submit the project

Features

- **Income and Expense Tracking:**
  Add, view, update, and delete **records** of income and expenses.
- **Loan and Investment Management:**
  Handle **special records** such as loans and investments with unique attributes.
  Loan is special income record and investment is special expense record
- **Budget Summary:**
  Display a summary of total income, total expenses, loans, and investments.
- **Unique Record Identification:**
  Use a static variable to assign unique IDs to each financial record.
- **User Interface:**
  A simple text-based menu for application navigation.

Learning Objectives

- Utilize basic C++ syntax and operations.
- Implement control structures for program flow.
- Understand and apply OOP concepts such as inheritance and polymorphism.
- Create and use complex data structures.
- Write functions for specific tasks and manage dynamic memory.
- Perform basic file I/O operations for data persistence.

Deliverables

- Source code files.
- A sample data file.

Assessment Criteria

- Correct implementation of specified features, including object-oriented designs.
- Code readability, including meaningful variable names, and comments.
- Functionality of the application, including handling of different record types.

This project is designed to be completed within a two-week timeframe and is aimed at reinforcing the students' understanding of both fundamental and advanced programming concepts in a practical, real-world application setting.

Sample Interaction

This section demonstrates a possible user navigation within the text-based menu of the "Personal Budget Manager" application, focusing on handling various financial records. This is just a sample and you are encouraged to be creative.

Main Menu

```
Welcome to the Personal Budget Manager!

Please choose an option:
1. Add Record
2. View Budget Summary
3. View All Records
4. Update Record
5. Delete Record
6. Exit

Enter your choice: [User inputs a number]
```

Adding Record

```
Enter your choice: 1

Select the type of record:
1. Income
2. Expense

Enter your choice: [User inputs a number for record type]
```

Adding Income Record

```
Select the type of income:
1. Earned Income
2. Loan

Enter your choice: [User inputs a number for income type]
```

Adding Earned Income

```
Enter the amount: 5000
Enter the date (YYYY-MM-DD): 2023-03-15
Enter a brief description: Salary
Income record added successfully!
Record ID: 0001
```

Adding Loan Record

```
Enter the loan amount: 10000
Enter the date (YYYY-MM-DD): 2023-04-01
Enter a brief description: Car Loan
Enter the interest rate (%): 5
Enter the borrower's name: John Doe
Enter the loan duration (years): 5
Loan record added successfully!
Record ID: 0002
```

Adding Expense Record

```
Select the type of expense:
1. Regular Expense
2. Investment

Enter your choice: [User inputs a number for expense type]
```

Adding Regular Expense

```
Enter the amount: 2000
Enter the date (YYYY-MM-DD): 2023-03-16
Enter a brief description: Groceries
```

```
Expense record added successfully!
Record ID: 0003
```

Adding Investment Record

```
Enter the investment amount: 15000
Enter the date (YYYY-MM-DD): 2023-04-15
Enter a brief description: Stock Investment
Enter the annual interest rate (%): 7
Enter the investment category: Stock
Investment record added successfully!
Record ID: 0004
```

Viewing Budget Summary

```
Enter your choice: 2

[Displays the total income, total expenses, and the net balance]
Total Income: $15000
Total Expenses: $7000
Net Balance: $8000
```

Viewing All Records

```
Enter your choice: 3

0001, Earned Income, $5000, 2023-03-15, Salary
0002, Loan, $10000, 2023-04-01, Car Loan, 5% Interest, John Doe, 5 Years
0003, Expense, $2000, 2023-03-16, Groceries
0004, Investment, $15000, 2023-04-15, Stock Investment, 7% Annual Interest, Stock
```

Updating a Record

```
Enter your choice: 4

Enter the record ID to update: [User inputs the record ID]
[User is prompted to enter new details for the selected record]
Record updated successfully!
```

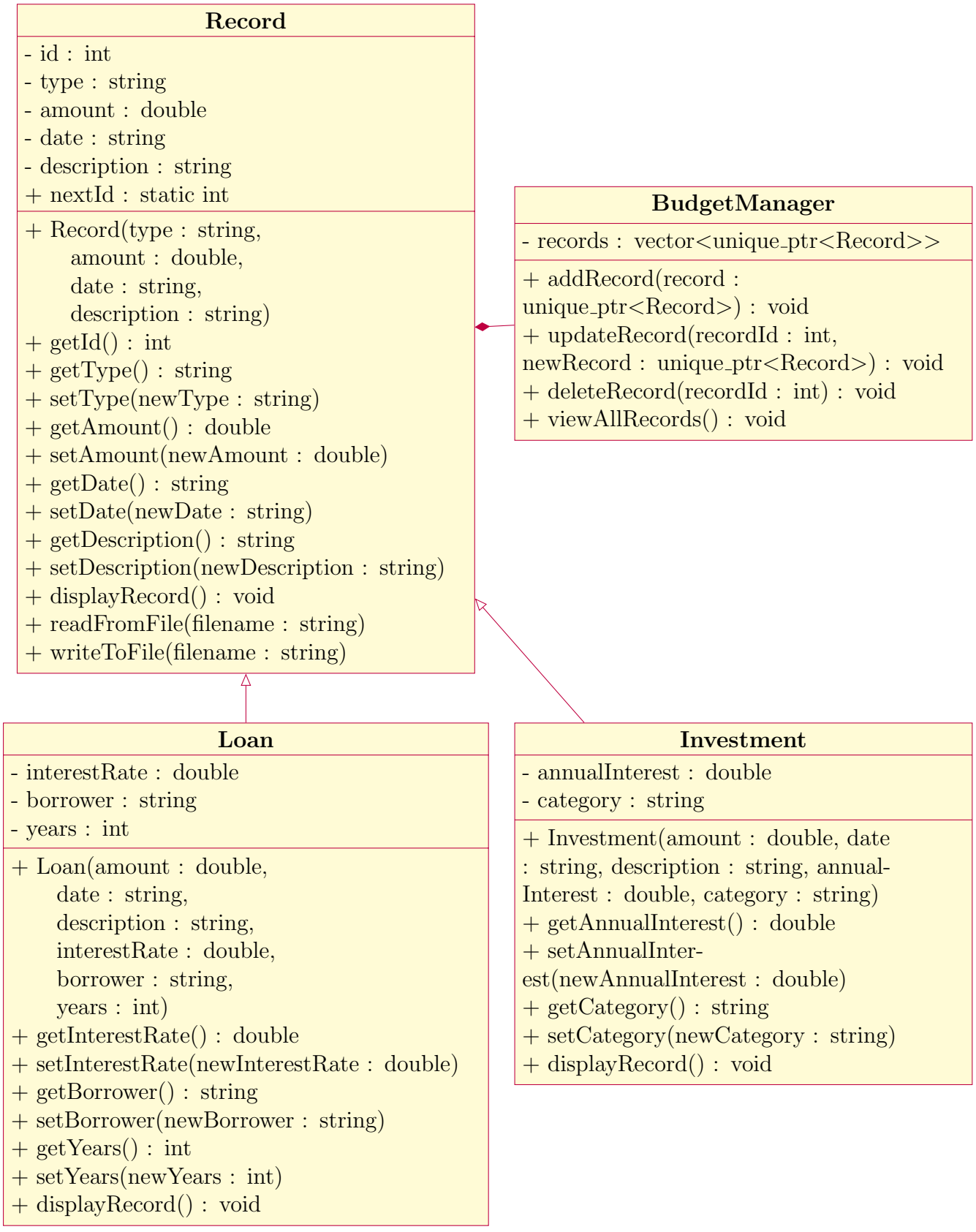
Deleting a Record

```
Enter your choice: 5
Enter the record ID to delete: [User inputs the record ID]
Record deleted successfully!
```

Exiting the Program

```
Enter your choice: 6
Thank you for using the Personal Budget Manager. Goodbye!
```

UML


Record
- id : int
- type : string
- amount : double
- date : string
- description : string
+ nextId : static int
+ Record(type : string, amount : double, date : string, description : string)
+ getId() : int
+ getType() : string
+ setType(newType : string)
+ getAmount() : double
+ setAmount(newAmount : double)
+ getDate() : string
+ setDate(newDate : string)
+ getDescription() : string
+ setDescription(newDescription : string)
+ displayRecord() : void
+ readFromFile(filename : string)
+ writeToFile(filename : string)
BudgetManager
- records : vector<unique_ptr<Record>>
+ addRecord(record : unique_ptr<Record>) : void
+ updateRecord(recordId : int, newRecord : unique_ptr<Record>) : void
+ deleteRecord(recordId : int) : void
+ viewAllRecords() : void
Loan
- interestRate : double
- borrower : string
- years : int
+ Loan(amount : double, date : string, description : string, interestRate : double, borrower : string, years : int)
+ getInterestRate() : double
+ setInterestRate(newInterestRate : double)
+ getBorrower() : string
+ setBorrower(newBorrower : string)
+ getYears() : int
+ setYears(newYears : int)
+ displayRecord() : void
Investment
- annualInterest : double
- category : string
+ Investment(amount : double, date : string, description : string, annualInterest : double, category : string)
+ getAnnualInterest() : double
+ setAnnualInterest(newAnnualInterest : double)
+ getCategory() : string
+ setCategory(newCategory : string)
+ displayRecord() : void