Object Oriented Programming

Design of Program I

2023/24

Lecture Outline

- What do we know?
- Object-oriented design of program
- Example

What do we know?

Class

- The class is a description of objects with common behavior.

```
class <name>
    private:
        <private members>
    public:
        <public members>
```

- Class members can be variables (data) and methods (functions).

Object

- The object is an instance – memory representation - of a class.
- The object is a representation of an entity that has a state represented by data and behavior represented by methods.

Constructor

- The constructor initializes objects.
- It has no return value.
- If not declared, the constructor is automatically generated (default, implicit constructor).
- It is called automatically when a static declaration or using **new**.
- *There may be more constructors, and they have to differ in number or type of parameters.*

Destructor

- Used for deallocation of memory created (allocated) dynamically.
- It has no return value.
- If it is not declared, a (default) destructor is automatically generated.
- It is used automatically by using the **delete** keyword.

Object-oriented design

Assignment

- Consider a small bank with a limited number of clients and accounts. The number of clients and accounts in the bank may increase.
- Each account has one owner and may have one partner, both are clients of the bank and have name and code. We can deposit, withdraw and check the status of an account. If there is not enough money in the account, we are not able to withdraw.
- Individual accounts have an interest rate, either basic or specific. Once upon a time, amount based on the corresponding interest rate is added to all bank accounts.
- Accounts and clients can be searched by their numbers or codes.
- We can get, but can not change:
    - account number and balance
    - code and the name of the client
    - owner and partner of an account

Classes

- Consider a small **bank** with a limited number of **clients** and **accounts**. The number of clients and accounts in the bank may increase.
- Each account has one owner and may have one partner, both are clients of the bank and have name and code. We can deposit, withdraw and check the status of an account (balance). If there is not enough money in the account, we are not able to withdraw.
- Individual accounts have an interest rate, either basic or specific. Once upon a time, amount based on the corresponding interest rate is added to all bank accounts.
- Accounts and clients can be searched by number or code.

Behavior

- Consider a small bank with a limited number of clients and accounts. The number of clients and accounts in the bank may **increase**.
- Each account has one owner and may have one partner, both are clients of the bank and have name and code. We can **deposit**, **withdraw** and **check the status** of an account (balance). If there is not enough money in the account, we are not able to withdraw.
- Individual accounts have an interest rate, either basic or specific. Once upon a time, amount based on the corresponding interest rate **is added** to all bank accounts.
- Accounts and clients **can be searched** by number or code.

State

- Consider a small bank with a limited number of clients and accounts. The number of **clients** and **accounts** in the bank may increase.

- Each account has **one owner** and may have **one partner**, both are clients of the bank and have **name** and **code**. We can deposit, withdraw and check the **status of an account (balance)**. If there is not enough money in the account, we are not able to withdraw.

- Individual accounts have an **interest rate**, either basic or specific. Once upon a time, amount based on the corresponding interest rate is added to all bank accounts.

- Accounts and clients can be searched by number or code.

Class *Client*

- Code and name
- We can get them
- We are not able to change them

Class *Account*

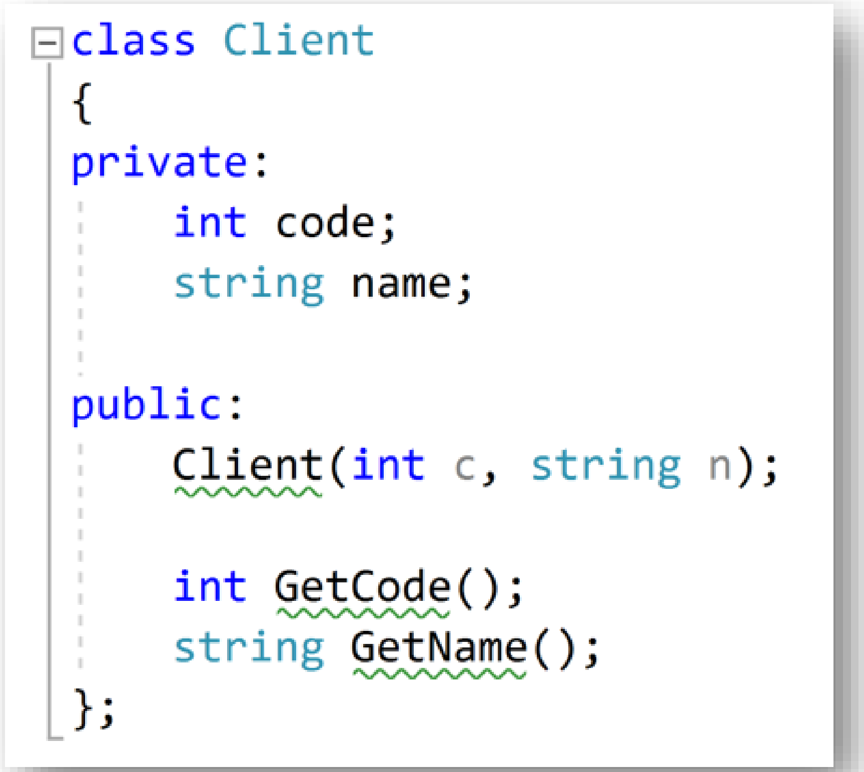
- Number and balance, owner and partner, interest rate.
- We can get them
- We are not able to change number, interest rate, owner and partner
- Deposit, withdraw, check state, add interest
- We cannot withdraw in some situations

Class *Bank*

- A limited list of clients and accounts.
- We can add new clients and accounts.
- We can add interest to all bank accounts according to their interest rates.
- We can search clients by code and accounts by numbers

Example

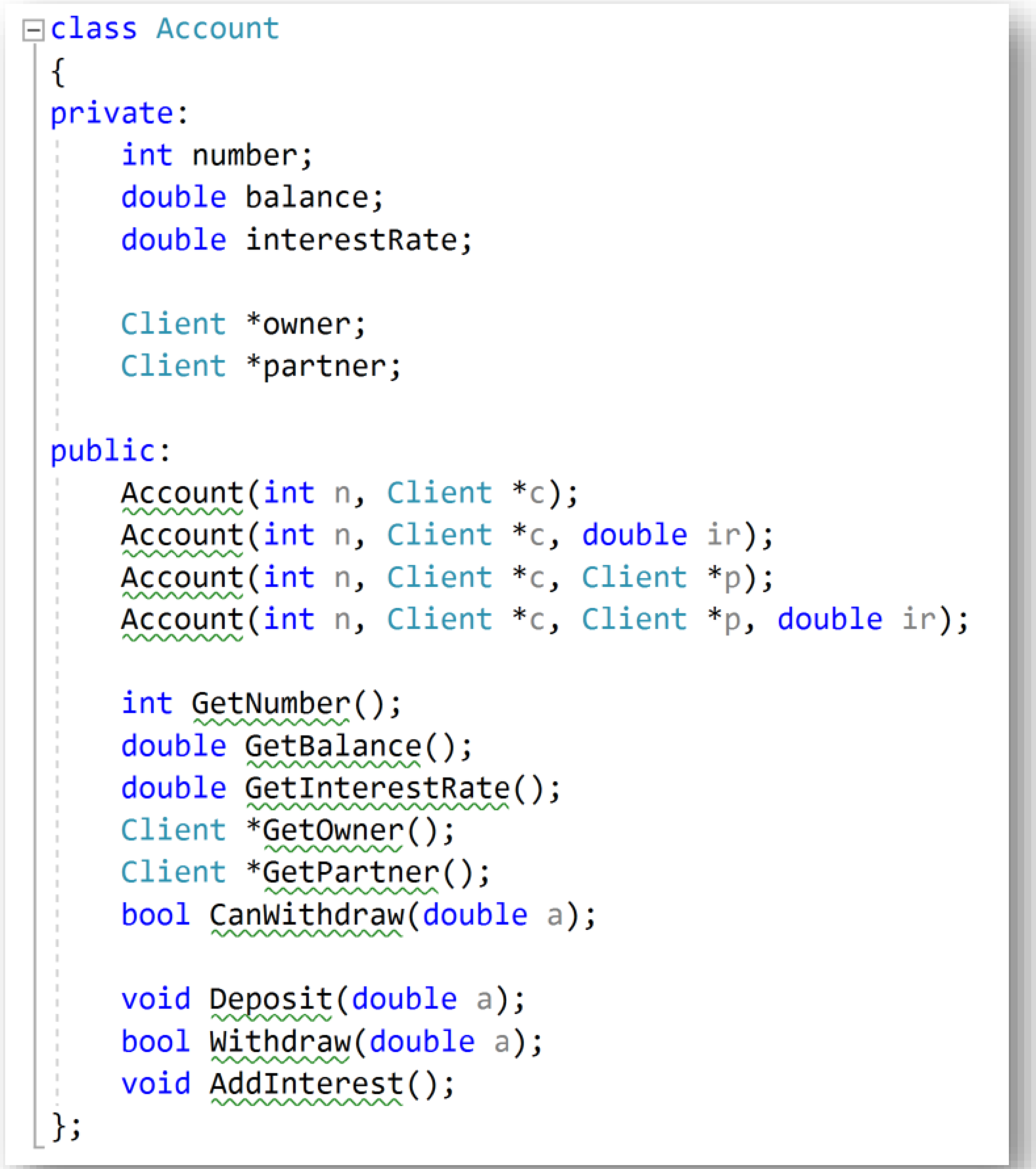
Declaration



```cpp
class Client
{
private:
    int code;
    string name;

public:
    Client(int c, string n);

    int GetCode();
    string GetName();
};
```



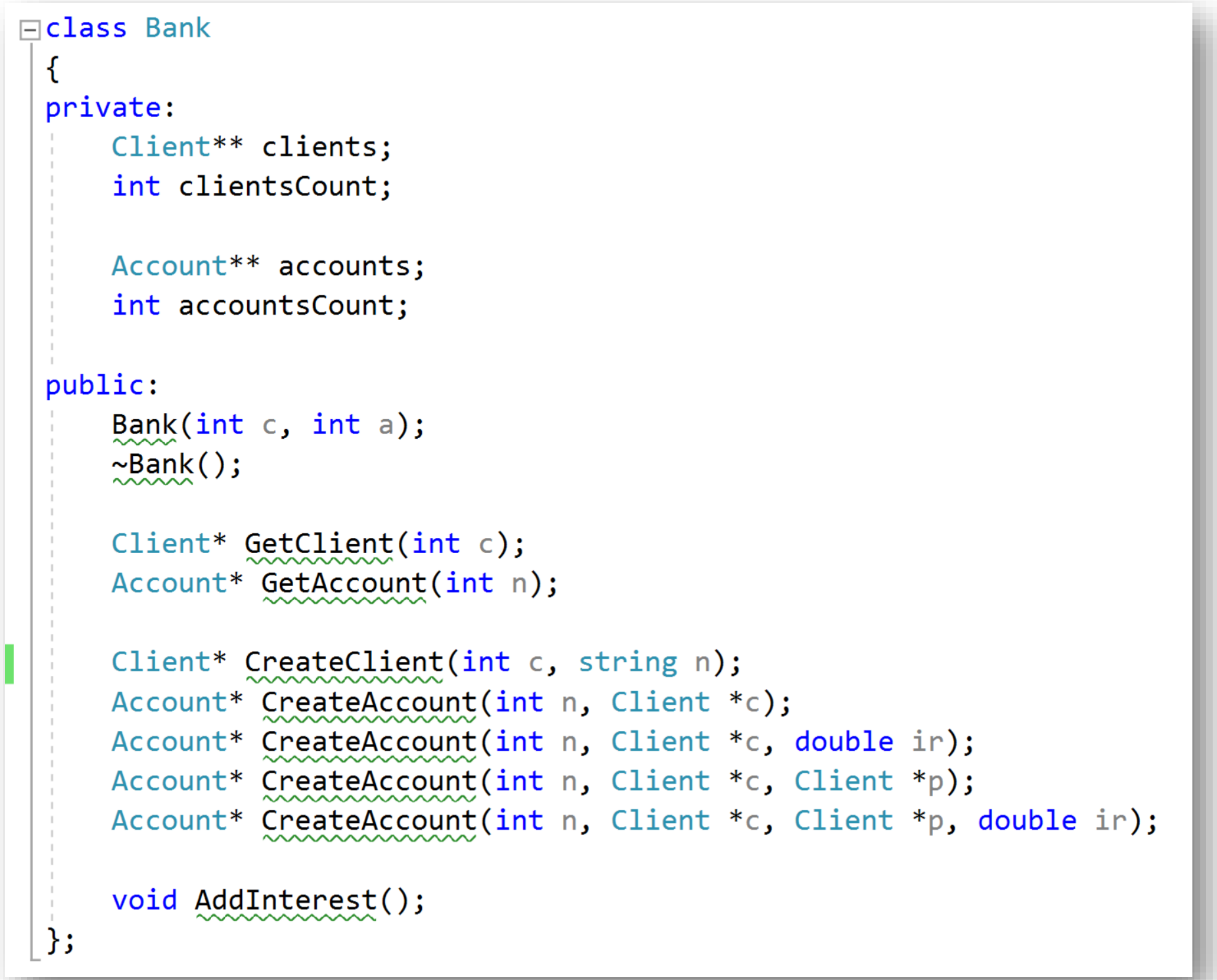
```
class Account
{
private:
    int number;
    double balance;
    double interestRate;

    Client *owner;
    Client *partner;

public:
    Account(int n, Client *c);
    Account(int n, Client *c, double ir);
    Account(int n, Client *c, Client *p);
    Account(int n, Client *c, Client *p, double ir);

    int GetNumber();
    double GetBalance();
    double GetInterestRate();
    Client *GetOwner();
    Client *GetPartner();
    bool CanWithdraw(double a);

    void Deposit(double a);
    bool Withdraw(double a);
    void AddInterest();
};
```



```
class Bank
{
private:
    Client** clients;
    int clientsCount;

    Account** accounts;
    int accountsCount;

public:
    Bank(int c, int a);
    ~Bank();

    Client* GetClient(int c);
    Account* GetAccount(int n);

    Client* CreateClient(int c, string n);
    Account* CreateAccount(int n, Client *c);
    Account* CreateAccount(int n, Client *c, double ir);
    Account* CreateAccount(int n, Client *c, Client *p);
    Account* CreateAccount(int n, Client *c, Client *p, double ir);

    void AddInterest();
};
```

Three types of methods

- **Constructors and destructor.** The constructors initialize the state of the object after its creation; the destructor releases the dynamically allocated memory before the object destruction. If not listed, these methods are created automatically (with no algorithm).
- **Methods providing information about the state of the object.** The methods either provide information directly about a value of the object data member or provide information based on an algorithm.
- **Methods changing object state.** The methods either directly change the value of the object data member or make a change based on an algorithm.
- *Methods providing information about the state of the object should not change its state!!!*

Object compositions

- The object can become part of another object and becomes its data member.
- This principle creates complex objects with well-defined competencies. These, however, can be realized through the interaction of the objects from which they are composed.
- For example, an *account* has *clients* in the roles of owner and partner, respectively. The *bank* has *clients* and *accounts*.
- The same object can be part of multiple compositions. For example, one *client* can be part of both an *account* and a *bank*.

Seminar assignments

- Implement the example from the lecture and design a code that will use all classes. Create dozens of clients and accounts of the bank and simulate some common tasks performed at the bank.

- Design and implement a similar system such as, for example, a medical office, a small school, etc.

Seminar Questions

- Explain how objects are created, what is a constructor and how to use it in C ++.
- Explain how objects are destroyed, what is a destructor how to use it in C ++.
- Explain the difference between static and dynamic declarations of objects in C ++.
- How can we find classes, their methods and data members in an assignment?
- When and why do we need to use multiple constructors of one class?
- When and why we need to declare and define destructors?
- What are default constructors and destructors and when do we need them?
- What kinds of methods do we usually have to implement?
- What are object compositions and why use them?