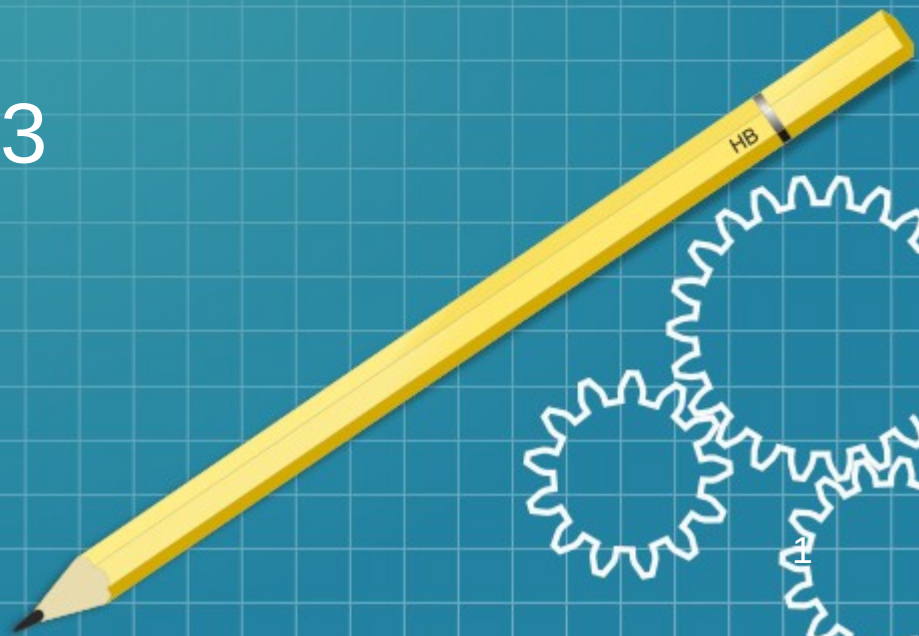
Introduction to Classes, Objects, Member Functions and Strings

Chapter 3



Objects

- In real world we are surrounded by objects.

Objects

- In software development, use of *objects*, or *classes* the objects come from, gives us:
  - easier to understand code
  - easier to support/debug/modify code
  - code re-useability

Classes

- In C++ *objects* are *instantiated* from *classes*

A simple analogy from real-world: a watch is made from its engineering drawings.

Classes

- In C++ *objects* are *instantiated* from *classes*

An *object* is an *instance* of its *class*.

Classes

A *class* has:

- *Attributes* / *data members*
- *Messages* / *member-functions*

Account class

Account
- name: string - balance: int
Account(accountName: string, initialBalance: int) + deposit(depositAmount : int) + getBalance() : int + setName(accountName: string) + getName(): string

Account class

```
#include <string>

class Account {

public:

   void setName(std::string accountName){
      name = accountName;
   }

   std::string getName() const {
      return name;
   }

private:
   std::string name;

};
```

Account class

```
#include <string>

class Account {
public:
   void setName(std::string accountName){
      name = accountName;
   }

   std::string getName() const {
      return name;
   }

private:
   std::string name;
};
```

We will use C++ string data type

access specifiers (keywords)

accessible to any function/ member-function

accessible only to Action's member-functions

Account class

- *Access specifiers*:
  - `private`

    accessible only to member-functions of this class
  - `public`

    accessible to any function / member-function outside this class
  - `protected`

    accessible by members and friends of this class, and by members and friends of classes derived from this class

Account class

```
#include <string>

class Account {

public:

   void setName(std::string accountName){
      name = accountName;
   }

   std::string getName() const {
      return name;
   }

private:
   std::string name;

};
```

member-function, sets the account holder's name in the object

member-function, returns the account holder's name

data member, contains account holder's name

Account class

```
#include <string>

class Account {

public:

   void setName(std::string accountName){
      name = accountName;
   }

   std::string getName() const {
      return name;
   }

private:
   std::string name;

};
```

What is missing in our implementation?

Account class

Account
- name: string - balance: int
Account(accountName: string, initialBalance: int) + deposit(depositAmount : int) + getBalance() : int + setName(accountName: string) + getName(): string

Account class

Account
- name: string - balance: int
Account(accountName: string, initialBalance: int) + deposit(depositAmount : int) + getBalance() : int + setName(accountName: string) + getName(): string

Account class

Account
- name: string ✓ - balance: int
Account(accountName: string, initialBalance: int) + deposit(depositAmount : int) + getBalance() : int + setName(accountName: string) ✓ + getName(): string ✓

Let's go ahead and add the two missing member functions and constructor:

- download the `Account.h` from our web-site
- open it in an editor

Account class

Now we have:

- `Account2.h` : definition of the Account class
- `AccountTest2.cpp` : code that uses the Account class

In-class work

Exercise 3.9 (Modified `Account` Class):
Modify the `Account` class to provide a *member function* called `withdraw` that withdraws money from an `Account` object.

Ensure that the withdrawal amount does not exceed the `Account`'s balance. If it does, the balance should be left unchanged and the member function should display a message indicating "*withdrawal amount exceeded account balance*".

Modify the `AccountTest2.cpp` to test the member function `withdraw`.

In-class work

Exercise 3.14 (C++ 11 *List Initialization*):
Write a statement that uses *list initialization* to initialize object of class `Account` which provides a constructor that receives an `unsinged int`, two `strings` and a `double` to initialize the `accountNumber`, `firstName`, `lastName` and `balance` data members of new object of the class.

In-class work

Exercise 3.12 (`Date` class):
Create a class called `Date` that includes three pieces of information:

- a month (type int)
- a day (type int)
- a year (type int).

Your class should have a *constructor* with three parameters to initialize the three *data members*.
Assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1 – 12; if it isn't, set the month to 1.
Provide a `set` and a `get` functions for each data member.
Provide a member function `displayDate` that displaysthe month, the day and the year separated by /.
Write a test program that demonstrates class `Date`'s capabilities.

HW assignment

1) Exercise 3.10
2) Exercise 3.13
3) more will be posted from Chapter 4 in the next class

Suggested exercises
(*not for grade, but the questions related to these will appear on a quiz or a test*):
1) Chapter 3,
   Summary and all Self-Review Exercises (pages 95-100) -
2) Chapter 3, Exercises (page 100): 3.5 - 3.8
3) more will be posted from Chapter 4 in the next class