Week 11

Intro to Computer Science - CSI 1430

Hello and welcome to the weekly resources for CSI 1430: Introduction to Computer Science! This week's resource will be covering the course material that will be covered in week 11 of the course.

Last week we introduced classes, so this resource will be an expansion off of that. If you have not already...go back and quickly look through that resource as this week we will pick up exactly where we left off. **Just as a reminder:** These documents are intended for going over concepts that are discovered in class, as they are a concise and to the point review of the materials. But remember, these documents are not a replacement for lecture and they cannot cover everything, so make sure you get help on any specific concepts you struggle with as well as go over lecture notes, programs, and the textbook.

Reminder: If you have any questions about these study guides, group tutoring sessions, private 30 minutes tutoring appointments, the Baylor Tutoring Youtube channel or any tutoring services we offer, please visit our website https://baylor.edu/tutoring or call our drop in center during open business hours. Monday-Thursday 9am - 8pm on class days (254) 710-4135.

Keywords: *Classes, Class Functionality*

TOPIC OF THE WEEK

Classes

Last week we talked about how we can establish a class in C++. To recall, a class represents an object or an entity that has some defined attributes and actions that can be contained within it. The example we have been using is a Bank Account class, and we will continue to use this example to explore the other things available within classes.

```cpp
#include <iostream>
#include <string>
using namespace std;

class Bank_Account {

private:
    int account_number;
    string account_type;
    int amount;

public:
    Bank_Account();
    Bank_Account(int, string, int);

    void setAccNumber(int accID) { account_number = accID; };
    void setAccType(int accType) { account_type = accType; };
    int getAccNumber() { return account_number; };
    string getAccType() { return account_type; };
    int getAccAmount() { return amount; };

    void amount_withdraw(int);
    void amount_deposit(int);
    void account_closer();
};
```

For this, let's adjust our declaration of the Bank_Account class. The main differences to note are the addition of the amount attribute and the addition of a few set and get functions that we will discuss.

First of all, we need to discuss class access. There is a difference between public and private access within a class that basically acts as the type of restriction that non-class members will have to those sections of the class. Public means that those functions and attributes are accessible from outside of the class, whereas private means that they are not. There is also a protected access specifier, but this will not be useful yet. In our example, all of our actual class attributes are labeled under the private access specifier. Using this private label is what is known as data hiding. Data hiding guarantees restricted access to object members and maintains object integrity. Basically what is going on is that we are making access to the class data members by anything other than the actual class instance restricted directly. Rather, we will invoke the use of setter and getter methods to make sure we define the behavior for accessing and modifying these values. Overall being safer

Now in terms of things that we label public, this will be available to everything outside the class. Since we are able to define the behavior of functions, we can make sure access and modification is done in a correct and safe way. Setter and getter functions essentially do the same thing as normal access otherwise! Setter functions are able to change the value of an attribute by passing the new value as a parameter to the function. Getter functions take the stored value of the attribute for the object instance you are referring to and returns it back to you.

Now let's take a look at the other functions that are a part of the Bank_Account class. The type of functions left to discuss are the constructors and the general class methods.

A constructor is a method that's name matches the name of the class. A constructor is how an object is "constructed" when you declare a new object of

```
Bank_Account::Bank_Account() {
    account_number = -1;
    account_type = "NEW";
    amount = 0;
}

Bank_Account::Bank_Account(int accID, string accType, int accAmt) {
    account_number = accID;
    account_type = accType;
    amount = accAmt;
}

void Bank_Account::account_closer() {
    account_number = -1;
    account_type = "CLOSED";
    amount = 0;
}

void Bank_Account::amount_deposit(int addAmount) {
    amount += addAmount;
}

void Bank_Account::amount_withdraw(int subAmount) {
    amount -= subAmount;
}
```

that class type and how the initialization of that object is configured. Constructors can either be default which pass no parameters, or they can pass parameters. In our example, the default constructor is initializing the account to have an ID of -1, a type of “NEW”, and an amount of 0. Those values represent default values and essentially get the object “ready for use”. For other constructors, you can choose what variables you want passed as parameters. Doing this, you are able to mix default values and passed values to a class constructor.

The other class methods are available to perform some action internally to the class. Methods such as these that are part of the class include some sort of calculation and serve as the class behavior. In our Bank Account example, we have methods to deposit, withdrawal, and close the account. However, there is so much that these behavioral class functions can do based on what you are trying to achieve.

To wrap up our overview of classes, I think it's important to go over how they are declared and how functions are called and such.

To quickly explain what is happening here, basically we are creating a Bank_Account object called newAccount...With that object we are calling a setter method that is responsible for altering the account number. We also do a couple deposits and withdrawals that affect the standing total of the account balance.

```
int main() {

    Bank_Account newAccount;

    //Example of getter method
    cout << "Before Set ID: " << newAccount.getAccNumber() << endl;

    //Example of setter method
    newAccount.setAccNumber( accID: 12345);

    //Example of getter method (After SET call)
    cout << "After Set ID: " << newAccount.getAccNumber() << endl << endl;


    //Before deposit
    cout << "Before Deposit: " << newAccount.getAccAmount() << endl;

    //Example of setter method
    newAccount.amount_deposit( addAmount: 100);
    newAccount.amount_deposit( addAmount: 100);
    newAccount.amount_withdraw( subAmount: 20);
    newAccount.amount_deposit( addAmount: 100);

    //After deposit
    cout << "After Deposit: " << newAccount.getAccAmount() << endl;

    return 0;
}
```

Take a look through the code, especially the before and after, and take a look at what is output as a result of all the class function calls. The output is displayed to the right. Step through the program including the constructor and class calls.

```
Before Set ID: -1
After Set ID: 12345

Before Deposit: 0
After Deposit: 280
```

CHECK YOUR LEARNING

1. What are the 2 different types of access specifiers?
2. What should the name of the default constructor be for a class?
3. True or False: It's best to make all attributes for a class private and have them accessed by setter and getter methods.

THINGS STUDENTS MAY STRUGGLE WITH

1. The best way to practice and start understanding classes is to take a random everyday object (like a Person, Circle, House, etc) and try to define a list of attributes and behaviors for that object. From there you can write a simple class based on that object including those attributes, setters/getters, and a couple basic behaviors.
    - For example, I can take a Person and make a class with attributes of age, height, eye color, and social security number. With those, I can create setters and getters for the attributes along with a couple different behavior functions.
2. Syntax is super important for classes! Take some time to get familiar with creating simple classes through repetition because the concept of a class will never go away through your computer science education.

Thanks for checking out these weekly resources! Don't forget to check out our website for group tutoring times, video tutorials and lots of other course resources: https://baylor.edu/tutoring. The answers to the 'Check Your Learning' questions are below.

Answers:

1. Public and Private
2. The same as the name of the class
3. True, it's a better practice and safer to have your classes perform data hiding than to allow direct access.

Note: All tables were taken from the Baylor CS 1430 zyBook