C++ Primer Plus, 5^{th} *Edition* by Stephen Prata
Chapter 10: Objects and Classes
Review Questions

1. What is a class?

   A class is a user-defined type that is an abstraction define in terms of its interface with the user. It allows the user to access to member variables and member functions appropriate for the type all in one place. In a general sense, a class is a template for an object that provides a set of instructions for the data the object will hold and the operations that can be performed on the object.

2. How does a class accomplish abstraction, encapsulation, and data hiding?

   Abstraction is accomplished by how the functions are defined in terms of what the user can do with the object. Encapsulation is accomplished by how the member variables and member functions are all in one place. Data hiding is accomplished by the private an public sections of a class, where private data is only accessable by the object and cannot be accessed directly by the user.

3. What is the relationship between an object and a class?

   An object is an instance of a class. In a general sense, a class is a template for the data that an object will hold and the operations that can be perfomed on the object. The object is the actual variable that the user deals with directly. As an analogy, a class is a blueprint for a building, and an object is the physical building after construction.

4. In what way, aside from being functions, are class function members different from class data members?

   Functions of classes can access private data of the same class. Generally, functions are how the user interacts with the data members of an object, because it is good programming style to keep data members of an object private.

5. Define a class to represent a bank account. Data members should include the depositor's name, the account number (use a string), and the balance. Member functions should allow the following.
    - Creating an object and initializing it.
    - Displaying the depositor's name, account number, and balance.
    - Dspositing an amount of money given by an argument
    - Withdrawing an amount of money given by an argument

    Just show the class declaration, not the method implementations. (Programming Exercise 1 provides you with an opportunity to write the implementation.)

    See the following code:

```
class BankAccount
{
private:
    string name;
    string accountNum;
    double balance;
public:
    BankAccount(string name, string account, double balance);
    void PrintAccount();
    void Deposit(double sum);
    void Withdraw(double sum);
    ~BankAccount();
};
```

6. When are class constructors called? When are class destructors called?

    Class constructors are called upon declaration of the object. A constructor is what allows an object to come into existance. Destructors are called implicitly and depend on the type of storage class for the object. For example, if the object is static, the destructor will be called when the program terminates. If the object is a local variable, the destructor will be called when the program leaves the block of code that the object was declared in.

7. Provide code for a constructor for the bank account class from Review Question 5.

    See the following code:

```
BankAccount::BankAccount(string name, string account, double balance)
{
    this->name = name;
    this->accountNum = account;
    this->balance = balance;
    return;
}
```

8. What is a default constructor? What is the advantage of having one?

   The default constructor is a constructor that is called implicitly when the object is declared in the event that the user did not create his own constuctor. The advantage of having one is that it makes the use of objects more like the other fundamental data types. Also, less code needs to be written and the behavior of a default constructor is very well defined.

9. Modify the `Stock` class (the version in `stock2.h`) so that it has member functions that return the values of the individual data members. Note: A member that returns the company name should not provide a weapon for altering the array. That is, it can't simply return a `char *`. It could return a `const` pointer, or it could return a pointer to a copy of the array, manufactured by using `new`.

   See the following code:

```
#ifndef STOCK2_H_
#define STOCK2_H_

class Stock
{
private:
    char company[30];
    int shares;
    double share_val;
    double total_val;
    void set_tot() {total_val = shares * share_val;}
public:
    Stock();     // Default constructor
    Stock(const char * co, int n = 0, double pr = 0.0);
    ~Stock();    // Do-nothing destructor
    void buy(int num, double price);
    void sell(int num, double price);
    void update(double price);
    void show() const;
    const Stock & topval(const Stock & s) const;

    // functions to return values of data members
    const char * showCompany() {return this->company;}
    int showShares() {return shares;}
    double showShareValue() {return share_val;}
    double showTotalValue() {return total_val;}
};

#endif
```

10. What are `this` and `*this`?

    `this` *is a pointer to the object that* `this` *is used in.* `*this` *is the object.*