Activity #23: Inheritance
Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #4

b) Model 2, Question #7.d

c) Model 2, Question #10.a

Activity #23: Inheritance

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to inheritance in C++.

Content Learning Objectives

After completing this activity, students should be able to:

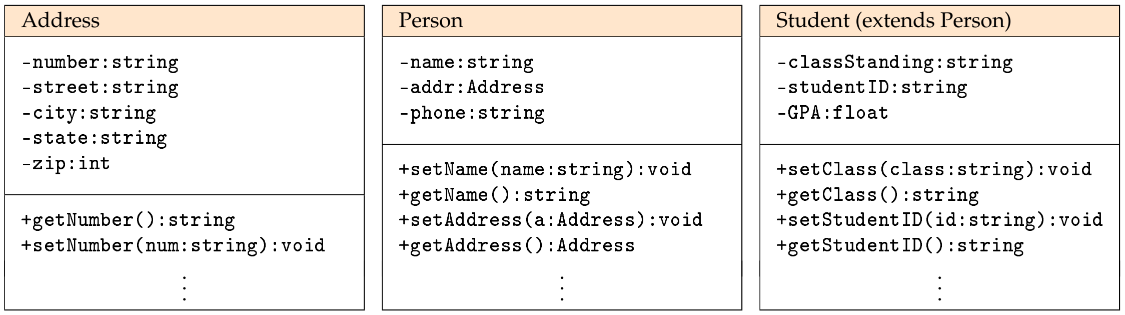
- Explain the difference between "Is-a" and "Has-a" relationships
- Explain how derived classes and base classes and their members are related

Process Skill Goals

During the activity, students should make progress toward:

- Write a member function which overrides the base class's function

Model 1 Several Classes from an University Information System



```
Address
-number:string
-street:string
-city:string
-state:string
-zip:int
+getNumber():string
+setNumber(num:string):void
⋮
```

```
Person
-name:string
-addr:Address
-phone:string
+setName(name:string):void
+getName():string
+setAddress(a:Address):void
+getAddress():Address
⋮
```

```
Student (extends Person)
-classStanding:string
-studentID:string
-GPA:float
+setClass(class:string):void
+getClass():string
+setStudentID(id:string):void
+getStudentID():string
⋮
```

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (25 min) Start time:

1. Large programs often contain many different classes. These classes will frequently be related to each other through "has-a" relationship, where one class has a data member that is itself an object of another class. Another less-common relationship is an "is a" relationship, where one class is a more general class, while the second is a more specialized version of the original class.

a) Which class is used as a data member type inside another class?

b) Is this an example of a "has-a" or an "is-a" relationship? Explain.

c) Which two classes have an "is-a" relationship? Explain.

d) Of the classes indicated above, which is the more general and which is the more specialized?

e) Would the member functions shown for the more general class be appropriate in the specialized class as well? Explain.

f) Give an example not shown in the model of a member function that might appear in the specialized class but not in the more general class.

2. Explain why the relationship between `Person` and `Address` is not an "is-a" relationship.

3. As a group, come up with another example of a "has-a" relationship. Fill in the information below about your example.

a) What class is used as a variable inside the other?

b) What class is the variable inside of?

c) Fill in the blanks below to describe the relationship between your two classes.

Every *has a(n)*

4. As a group, come up with another example of an "is-a" relationship. Fill in the information below about your example.

a) What class is the more general class?

b) What class is the specialized class?

c) Fill in the blanks below to describe the relationship between your two classes.

Every *is a(n)*

Model 2 Another "Is-A" Relationship

```cpp
// class for a deposit account
//   at a bank
class Account {
  public:
    Account(double balance);
    void deposit(double amount);
    virtual void withdraw(double amount);
    double getBalance() const {
      return balance;
    };
  private:
    double balance = 0;
};
```

```cpp
// class for a savings account
class Savings : public Account {
  public:
    Savings(double balance, double rate);
    void setRate(double rate);
    void newMonth();
    int getWithdrawals() const {
      return numWithdrawals;
    };
  private:
    double rate;
    int numWithdrawals;
};
```

Refer to Model 2 above as your group develops consensus answers to the questions below.

Questions (25 min) Start time:

5. In C++ the more general class in an "is-a" relationship is called the *base* class and the specific class is called the *derived* class. Answer the following questions about the model above.

a) What is the name of the base class?

b) What is the name of the derived class?

c) In C++ class definitions, how is the "is-a" relationship indicated?

6. An important property of derived classes is that they *inherit* the properties of the base class. Suppose that a new savings account was declared using the code: `Savings myAccount(50,0.01)`.

a) What data members does the object `myAccount` have?

b) In which class is each of these data members declared?

c) What member functions does the object `myAccount` have?

d) Which of these are inherited from the base class?

7. The file `activity22.cpp` contains a preliminary implementation of these two classes along with a sample `main` function. Use it to answer the following questions.

a) Without attempting to compile the code, describe what the member function `Savings::newMonth()` on line 94 is supposed to do.

b) Now compile the code and describe what happens.

c) Replace `this->balance` on line 85 with `this->getBalance()` Does this fix the problem?

d) Based on what you've observed in this question, does the derived class `Savings` have direct access to all of the members that it inherits from the base `Account` class? Explain.

8. The function `testAccount()` defined in `activity22.cpp` tests our implementation of these classes by depositing \$50 and then attempting to make ten \$10 withdrawals.

a) What are the results of running on this function on the checking account with an initial balance of \$100, as defined in the `main` program?

b) What are the results of running this function twice in successive months on the savings account with an initial balance of \$80, an interest rate of 1%?

c) What type is the `testAccount` function's single parameter?

d) What type is the argument passed to this function on line 52?

e) What type is the argument passed to this function on line 55?

f) Why is this worth noting?

9. In the United States, you are not allowed to make more than six withdrawals from a savings account in one month. Based on this new information, are any of the member functions that `Savings` inherits from `Account` not quite appropriate? Explain what is missing from these functions.

10. We can *override* a member function from the base class by defining a member function the same name, parameters, and return type in the derived class.

a) How is this different from *overloading* a function?

b) Define a `bool Savings::withdraw(double balance)` function that overrides the `withdraw()` function from the `Account` class and does not allow more than 6 withdrawals. Hint: You can use `Account::withdraw()` to call the base class `withdraw()` from within your new definition.

c) What did you have to add to the declaration of the `Savings` class?

d) Now run the program again. How did the results of the savings account test change?

e) What happens if you remove the keyword `virtual` from line 12?