Inheritance Basics

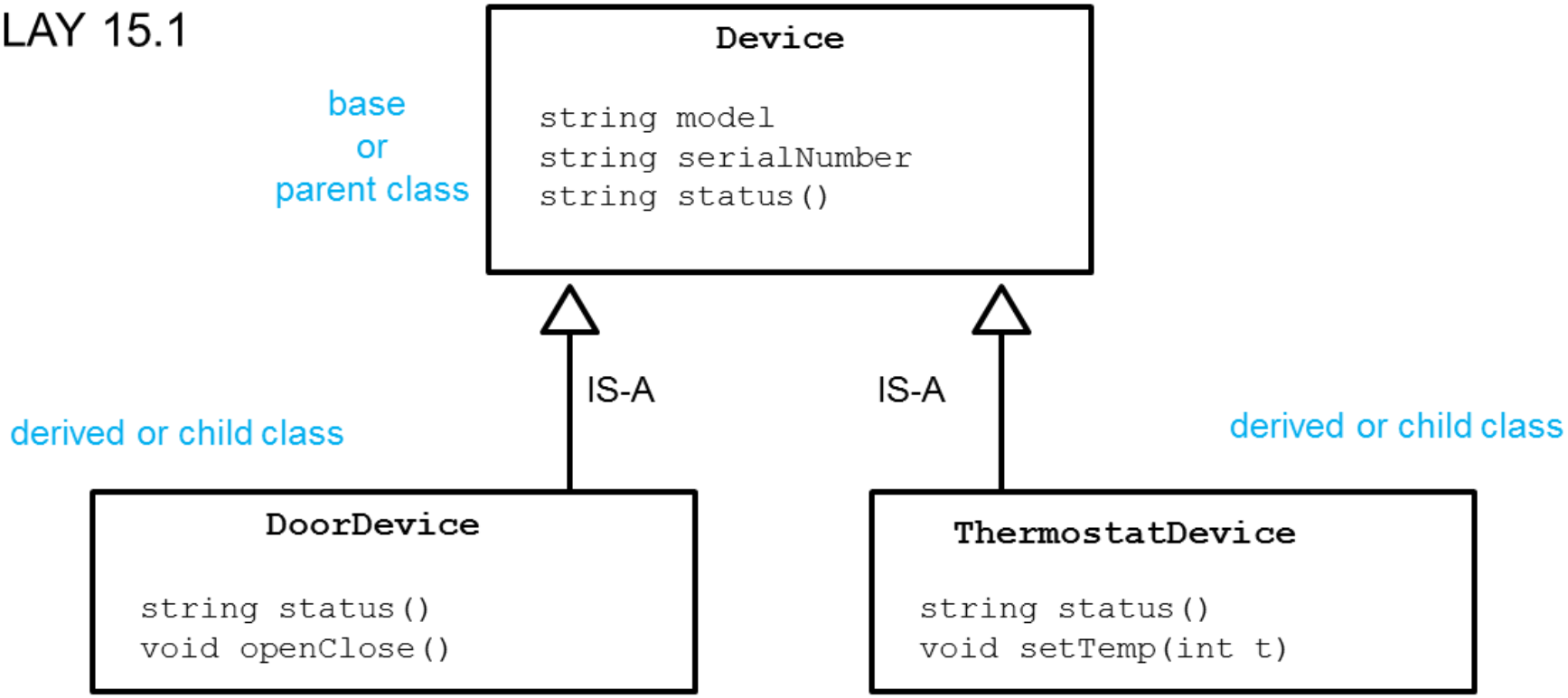
Modified from Section 15.1

Inheritance Basics

- Inheritance is the process by which a new class, called a derived class, is created from another class, called the base class
  - A derived class automatically has all the member variables and functions of the base class
  - A derived class can have additional member variables and/or member functions
  - The derived class is a child of the base or parent class

Example Inheritance Hierarchy for Home Automation Devices

DISPLAY 15.1



An object of type `DoorDevice` or `ThermostatDevice` includes functions and variables defined in `Device`, such as `model` and `serialNumber`.

Employee Classes

- To design a record-keeping program with records for salaried and hourly employees…
  - Salaried and hourly employees belong to a class of people who share the property "employee"
  - A subset of employees are those with a fixed wage
  - Another subset of employees earn hourly wages
- All employees have a name and SSN
  - Functions to manipulate name and SSN are the same for hourly and salaried employees

A Base Class

- We will define a class called Employee for all employees
- The Employee class will be used to define classes for hourly and salaried employees

- A definition of the employee class is found in **Display 15.2** **Display 15.3**

Interface for the Base Class Employee

Display 15.2

Back Next

```
//This is the header file employee.h.
//This is the interface for the class Employee.
//This is primarily intended to be used as a base class to derive
//classes for different kinds of employees.
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>
using namespace std;

namespace employeessavitch
{

    class Employee
    {
    public:
        Employee( );
        Employee(string the_name, string the_ssn);
        string get_name( ) const;
        string get_ssn( ) const;
        double get_net_pay( ) const;
        void set_name(string new_name);
        void set_ssn(string new_ssn);
        void set_net_pay(double new_net_pay);
        void print_check( ) const;
    private:
        string name;
        string ssn;
        double net_pay;
    };

}//employeessavitch

#endif //EMPLOYEE_H
```

Display 15.3 (1/2)

Implementation for the Base Class `Employee` (*part 1 of 2*)

```
//This is the file: employee.cpp.
//This is the implementation for the class Employee.
//The interface for the class Employee is in the header file employee.h.
#include <string>
#include <cstdlib>
#include <iostream>
#include "employee.h"
using namespace std;

namespace employeessavitch
{
    Employee::Employee( ) : name("No name yet"), ssn("No number yet"), net_pay(0)
    {
        //deliberately empty
    }

    Employee::Employee(string the_name, string the_number)
       : name(the_name), ssn(the_number), net_pay(0)
    {
        //deliberately empty
    }

    string Employee::get_name( ) const
    {
        return name;
    }

    string Employee::get_ssn( ) const
    {
        return ssn;
    }
```

Display 15.3 (2/2)

Implementation for the Base Class `Employee` (*part 2 of 2*)

```
double Employee::get_net_pay( ) const
{
    return net_pay;
}

void Employee::set_name(string new_name)
{
    name = new_name;
}

void Employee::set_ssn(string new_ssn)
{
    ssn = new_ssn;
}

void Employee::set_net_pay (double new_net_pay)
{
    net_pay = new_net_pay;
}

void Employee::print_check( ) const
{
    cout << "\nERROR: print_check FUNCTION CALLED FOR AN \n"
         << "UNDIFFERENTIATED EMPLOYEE. Aborting the program.\n"
         << "Check with the author of the program about this bug.\n";
    exit(1);
}

}//employeessavitch
```

Function print_check

- Function print_check will have different definitions to print different checks for each type of employee
  - An Employee object lacks sufficient information to print a check
  - Each derived class will have sufficient information to print a check

Class HourlyEmployee

- HourlyEmployee is derived from Class Employee
  - HourlyEmployee inherits all member functions and member variables of Employee
  - The class definition begins

    class HourlyEmployee : public Employee
    - :public Employee shows that HourlyEmployee is derived from class Employee
  - HourlyEmployee declares additional member variables wage_rate and hours

Display 15.4

Interface for the Derived Class HourlyEmployee

```
//This is the header file hourlyemployee.h.
//This is the interface for the class HourlyEmployee.
#ifndef HOURLYEMPLOYEE_H
#define HOURLYEMPLOYEE_H

#include <string>
#include "employee.h"

using namespace std;

namespace employeessavitch
{

    class HourlyEmployee : public Employee
    {
    public:
        HourlyEmployee( );
        HourlyEmployee(string the_name, string the_ssn,
                       double the_wage_rate, double the_hours);
        void set_rate(double new_wage_rate);
        double get_rate( ) const;
        void set_hours(double hours_worked);
        double get_hours( ) const;
        void print_check( ) ;
    private:
        double wage_rate;
        double hours;
    };

}//employeessavitch

#endif //HOURLYMPLOYEE_H
```

You only list the declaration of an inherited member function if you want to change the definition of the function.

Inherited Members

- A derived class inherits all the members of the parent class
  - The derived class does not re-declare or re-define members inherited from the parent, except…
  - The derived class re-declares and re-defines member functions of the parent class that will have a different definition in the derived class
  - The derived class can add member variables and functions

Implementing a Derived Class

- Any member functions added in the derived class are defined in the implementation file for the derived class
  - Definitions are not given for inherited functions that are not to be changed

- The HourlyEmployee class is defined in **Display 15.6**

Display 15.6 (1/2)

Implementation for the Derived Class `HourlyEmployee` (*part 1 of 2*)

```
//This is the file: hourlyemployee.cpp
//This is the implementation for the class HourlyEmployee.
//The interface for the class HourlyEmployee is in
//the header file hourlyemployee.h.
#include <string>
#include <iostream>
#include "hourlyemployee.h"
using namespace std;

namespace employeessavitch
{
    HourlyEmployee::HourlyEmployee( ) : Employee( ), wage_rate(0), hours(0)
    {
        //deliberately empty
    }

    HourlyEmployee::HourlyEmployee(string the_name, string the_number,
                                   double the_wage_rate, double the_hours)
    : Employee(the_name, the_number), wage_rate(the_wage_rate), hours(the_hours)
    {
        //deliberately empty
    }

    void HourlyEmployee::set_rate(double new_wage_rate)
    {
        wage_rate = new_wage_rate;
    }

    double HourlyEmployee::get_rate( ) const
    {
        return wage_rate;
    }
```

Display 15.6 (2/2)

Implementation for the Derived Class `HourlyEmployee` (*part 2 of 2*)

```
void HourlyEmployee::set_hours(double hours_worked)
{
    hours = hours_worked;
}

double HourlyEmployee::get_hours( ) const
{
    return hours;
}
```

We have chosen to set `net_pay` *as part of the* `print_check` *function since that is when it is used, but in any event, this is an accounting question, not a programming question. But note that C++ allows us to drop the* `const` *in the function* `print_check` *when we redefine it in a derived class.*

```
void HourlyEmployee::print_check( )
{
    set_net_pay(hours * wage_rate);

    cout << "\n_____________________________________________________\n";
    cout << "Pay to the order of " << get_name( ) << endl;
    cout << "The sum of " << get_net_pay( ) << " Dollars\n";
    cout << "_____________________________________________________\n";
    cout << "Check Stub: NOT NEGOTIABLE\n";
    cout << "Employee Number: " << get_ssn( ) << endl;
    cout << "Hourly Employee. \nHours worked: " << hours
         << " Rate: " << wage_rate << " Pay: " << get_net_pay( ) << endl;
    cout << "_____________________________________________________\n";
}
```

}//employeessavitch

Class SalariedEmployee

- The class SalariedEmployee is also derived from Employee
  - Function print_check is redefined to have a meaning specific to salaried employees
  - SalariedEmployee adds a member variable salary
- The interface for SalariedEmployee is found in **Display 15.5**

Display 15.7 (1-2) contains the implementation

Display 15.5

Interface for the Derived Class SalariedEmployee

```
//This is the header file salariedemployee.h.
//This is the interface for the class SalariedEmployee.
#ifndef SALARIEDEMPLOYEE_H
#define SALARIEDEMPLOYEE_H

#include <string>
#include "employee.h"

using namespace std;

namespace employeessavitch
{

    class SalariedEmployee : public Employee
    {
    public:
        SalariedEmployee( );
        SalariedEmployee (string the_name, string the_ssn,
                                    double the_weekly_salary);
        double get_salary( ) const;
        void set_salary(double new_salary);
        void print_check( );
    private:
        double salary;//weekly
    };

}//employeessavitch

#endif //SALARIEDEMPLOYEE_H
```

Display 15.7 (1/2)

Implementation for the Derived Class SalariedEmployee (*part 1 of 2*)

```
//This is the file salariedemployee.cpp.
//This is the implementation for the class SalariedEmployee.
//The interface for the class SalariedEmployee is in
//the header file salariedemployee.h.
#include <iostream>
#include <string>
#include "salariedemployee.h"
using namespace std;

namespace employeessavitch
{
    SalariedEmployee::SalariedEmployee( ) : Employee( ), salary(0)
    {
        //deliberately empty
    }

    SalariedEmployee::SalariedEmployee(string the_name, string the_number,
                                double the_weekly_salary)
                    : Employee(the_name, the_number), salary(the_weekly_salary)
    {
        //deliberately empty
    }

    double SalariedEmployee::get_salary( ) const
    {
        return salary;
    }

    void SalariedEmployee::set_salary(double new_salary)
    {
        salary = new_salary;
    }
```

Display 15.7 (2/2)

Implementation for the Derived Class SalariedEmployee (*part 2 of 2*)

```
    void SalariedEmployee::print_check( )
    {
        set_net_pay(salary);
        cout << "\n_____________________________________________________\n";
        cout << "Pay to the order of " << get_name( ) << endl;
        cout << "The sum of " << get_net_pay( ) << " Dollars\n";
        cout << "_____________________________________________________\n";
        cout << "Check Stub NOT NEGOTIABLE \n";
        cout << "Employee Number: " << get_ssn( ) << endl;
        cout << "Salaried Employee. Regular Pay: "
             << salary << endl;
        cout << "_____________________________________________________\n";
    }
}//employeessavitch
```

Parent and Child Classes

- Recall that a child class automatically has all the members of the parent class
- The parent class is an ancestor of the child class
- The child class is a descendant of the parent class
- The parent class (Employee) contains all the code common to the child classes
  - You do not have to re-write the code for each child

Derived Class Types

- An hourly employee is an employee
  - In C++, an object of type HourlyEmployee can be used where an object of type Employee can be used
  - An object of a class type can be used wherever any of its ancestors can be used
  - An ancestor cannot be used wherever one of its descendants can be used