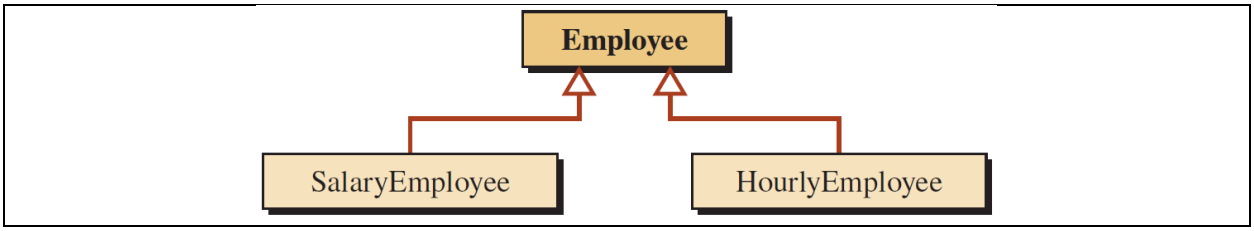
Object-Oriented Programming (3190)

Homework 4

Spring 2023

1. [10pt] Assume we have the following classes in a polymorphic relationship.



(a) The *Employee* class is an abstract class (only has pure virtual member functions). An employee has a first name(string), an initial(char), and a last name(string).

(b) A salary employee receives a fixed salary per month.

(c) An hourly employee receives wages based on the number of hours worked per month and a fixed rate/hour.

(d) All three classes must include *print* function.

(e) Write the interface files and the implementation files for all three classes and then test them using the given application file.

(f) Make sure that your output matches the following output and test your code with at least two more test cases on your own.

Possible Output:

```
Run:
Salary Employee:
John A. Pape
Salary : 2500
Payment: 2500

Hourly Employee:
Lucie C. Bush
Hours worked: 70
Rate: 20
Payment: 1400
```

Application File:

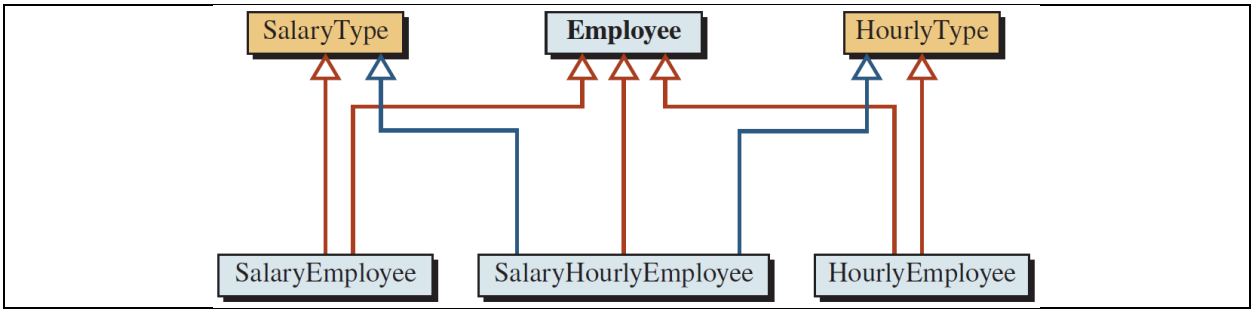
```
/*********************************************
 * The application file to instantiate and test classes
 * defined in the Shape hierarchy
 *********************************************/
#include "salaryEmployee.h"
#include "hourlyEmployee.h"

int main ()
{
        // Instantiation of Employee1 (first name, initial, last name, salary)
        SalaryEmployee employee1 ("John", 'A', "Pape", 2500);
        employee1.print();
        // Instantiation of Employee2 (first name, initial, last name, hours, rate)
        HourlyEmployee employee2 ("Lucie", 'C', "Bush" , 70, 20.0);
        employee2.print ();

        // Add your test code here:

        return 0;
}
```

2. [10pt] Modify the classes in question 1 with the following class relationships.



(a) Add another class named *SalaryHourlyEmployee* whose object receives a fixed amount of salary per month and extra pay if she works more than 180 hours per month using the same rate received by an hourly employee.

(b) Use two abstract classes (only have pure virtual member functions), *SalaryType* and *HourlyType*, to avoid multiple inheritance (mixin classes) as shown above.

(c) All six classes must include *print* function. For each print function of derived classes, use delegation from Employee's print function.

(d) Write the interface files and the implementation files for all classes and then test them using the given application file.

(e) Make sure that your output matches the following output and test your code with at least two more test cases on your own.

Possible Output:

```
Run:
John A. Pape
Salary Employee
Salary: 2500
Total Payment: 2500

Lucie C. Bush
Hourly Employee
Hours worked: 70
Rate: 20
Total Payment: 1400

Ann A. White
Salary Hourly Employee
Salary : 3500
Hours worked: 230
Rate: 20
Total Payment: 4500
```

Application File:

```
/************************************************
 * The application file to test all employee classes
 ************************************************/
#include "salaryEmployee.h"
#include "hourlyEmployee.h"
#include "salaryHourlyEmployee.h"

int main ()
{
        // Handling a salary employee (first name, initial, last name, salary)
        SalaryEmployee john ("John", 'A', "Pape", 2500);
        john.print();
        // Handling an hourly employee      (first name, initial, last name, hours, rate)
        HourlyEmployee lucie ("Lucie", 'C', "Bush" , 70, 20.0);
        lucie.print ();
        // Handling a salary-hourly employee (first name, initial, last name, salary, hours, rate)
        SalaryHourlyEmployee ann ("Ann", 'A', "White", 3500, 230, 20.0);
        ann.print ();

        // Add your test code here:

        return 0;
}
```

End of Assignment.