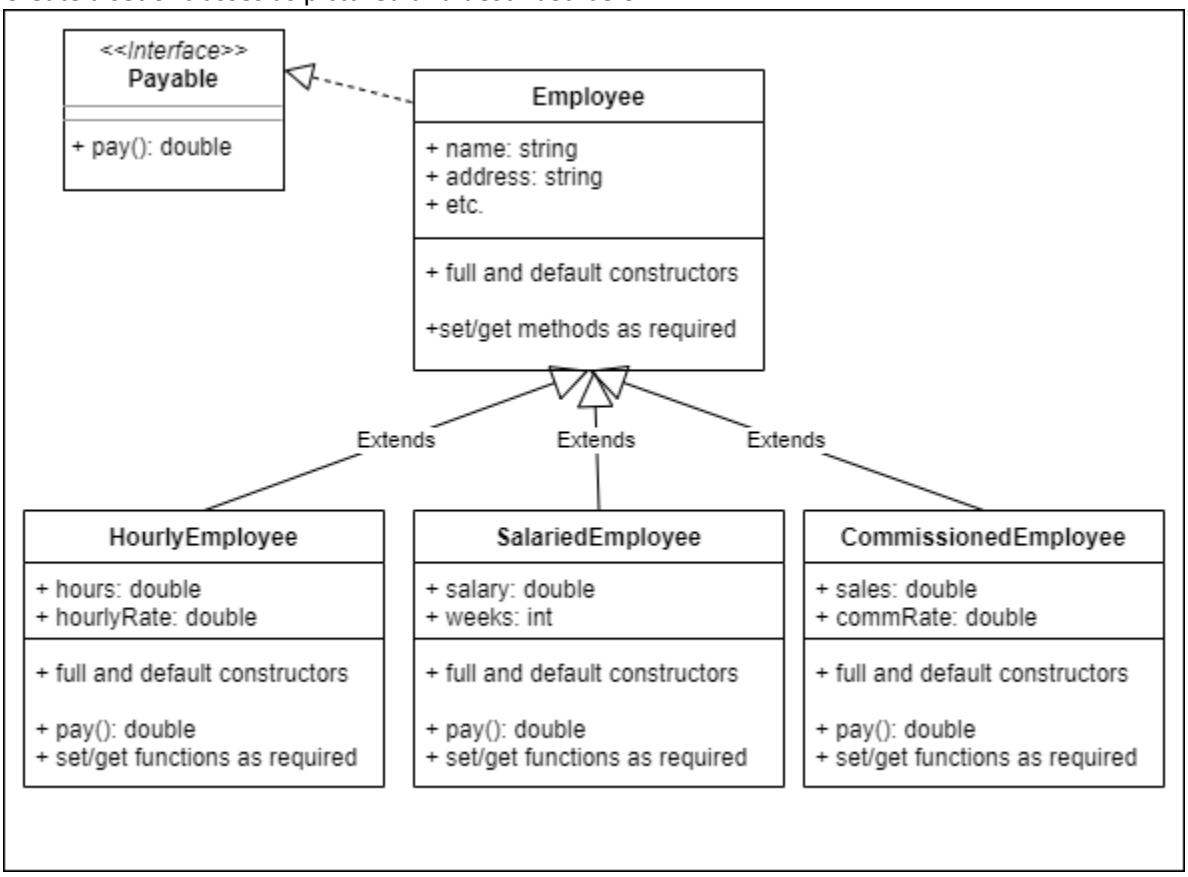
Assignment 5
Abstraction

F2021

Complete the project described below.

Classes

Create a set of classes as pictured and described below



- `Payable` is an interface
- `Employee` is an abstract class
- `HourlyEmployee`, `SalariedEmployee`, and `CommissionedEmployee` are "concrete" classes – which means instances can be made of them

`HourlyEmployee`	`SalariedEmployee`	`CommissionedEmployee`
`pay()` - hourlyRate * hours - sets hours to zero	`pay()` - salary * weeks - sets weeks to zero	`pay()` - commRate * sales - sets sales to zero

Make sure that all the constructors and set functions prevent inconsistent or invalid data. Example: an employee cannot have negative work hours (there are many others).

Assignment 5

F2021

Driver Program

Write a program that

- makes at least one instance of each of the concrete classes
- add all the instances to one array
- demonstrate that as you iterate through the array the `pay()` method works for each member in the manner it should

Technical Requirements

Write this program using header (.h) files and separate each class into its own pair of files.

Testing

Make sure that all of your code has been tested

- Every method in your code should be tested at least once.
- Remember when testing ranges of values to test above and below the allowable range, and exactly at the end points of the allowable range.

Style

Your program must be neat, well formatted, and readable (see the style guide on iLearn). Remember you can lose up to 30% for poor style. Don't forget comments that identify the programmer, the program, and the date the program was written.

Helpful idea?

Since there are some abstract classes here you might be tempted to wait until you have written a lot of code before you do any testing. I (Jim) would start with the base Employee class – without implementing the interface.

When I was confident that was working by building a mini-driver program that creates instances of Employee I would build one of the subclasses (still without implementing the interface) and test it.

When that is working I would go back and implement the interface on the base class – and then fix the subclass by writing the code required by the interface.

Then I would write the remaining subclasses (should be quick at this point).

Then I would delete (or move) the testing code I've been using and write the required driver file.