Lab 04 - Inheritance

Instructions:

- **A company operates as a hierarchical structure of individuals. Your task is to define a collection of related classes representing different roles within the company in a header file named '`Employee.h`'. For each class,**
  - ☐ **Define its special member functions.**
  - ☐ **Define getter and setter methods for its private fields.**
  - ☐ **Define a toString method that returns a string representation of the object.**
  - ☐ **Define a friend ostream operator (operator¡¡) for output formatting.**
  - ☐ **Define any other required methods as specified.**
  - ☐ **Make all fields private.**
  - ☐ **Make getter and toString methods take no parameters and be constant.**
  - ☐ **Define additional constructors and private methods if necessary.**
  - ☐ **Make all other methods and constructors public, unless explicitly stated otherwise.**
  - ☐ **Ensure that all methods adhere to the constraints and rules for the fields.**
- **'`Employee.h`' must contain a header guard.**
- **The classes must be defined within a namespace named '`oopl`'.**
- **'`Employee.h`' can only include the libraries *iostream*, *string*, *sstream*, *cstdlib*, *ctime*, *cctype*, *cmath*, and *iomanip*.**
- **Each method excluding special member functions, getter methods, and friends must include pseudocode as a comment above it to receive any credit.**
- **Your submissions must be submitted to the GitHub repository in the Lab04 directory.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating or failing to follow any of the rules above will result in an automatic zero (0) for the lab.**

Grading

Task	Maximum Points	Points Earned
1	1.25	
2	1.25	
3	1.25	
4	1.25	
Total	5.00	

Note: solutions will be provided for tasks colored blue only.

Task 1

- Define a class named *Person* such that
  - ☐ its fields, *firstname* and *lastname*, store only alphabetic characters and are initially set to `"john"` and `"doe"`, respectively. Additionally, all input values are automatically converted to lowercase.
  - ☐ *firstname* getter and setter methods are named `first()`.
  - ☐ *lastname* getter and setter methods are named `last()`.
  - ☐ the `toString()` method returns a string in the format

    "*x*, *y*"

    where *x* and *y* are capitalized *lastname* and *firstname*, respectively.

Task 2

- Define a class named *Employee* such that
  - ☐ it publicly inherits *Person*.
  - ☐ it defines a private static method named `genID()` that takes no parameters and returns a randomly generated string composed of four concatenated two-digit numbers.
  - ☐ its field, *identification*, is initialized to invocation of `genID()`.
  - ☐ *identification* getter method is named `number()` and it has no setter method.
  - ☐ the `toString()` method returns a string in the format

    "*n*\n*x*, *y*"

    where *n* is *identification*, and *x* and *y* are capitalized *lastname* and *firstname*, respectively.

Task 3

- Define a class named *HourlyEmployee* such that
  - ☐ it publicly inherits *Employee*.
  - ☐ its fields include a double named *payRate* and an integer named *weeklyHours*, with valid ranges of (3, 75] and (0, 4500], respectively. Their initial values are 18 for *payRate* and 2400 for *weeklyHours*.
  - ☐ *payRate* getter and setter methods are named `rate()`.
  - ☐ *weeklyHours* getter and setter methods are named `hours()`.
  - ☐ the `toString()` method returns a string in the format

    "*n*\n*x*, *y*\n*r* USD for *h* hours *m* minutes weekly"

    where *n* is *identification*, *x* and *y* are capitalized *lastname* and *firstname*, respectively, *r* is *payRate* with two decimal places, *h* is *weeklyHours* divided by 60, and *m* is *weeklyHours* modulo 60.

Task 4

- Define a class named *SalaryEmployee* such that
  - ☐ it publicly inherits *Employee*.
  - ☐ it has a double field named *pay*, with a valid range of [20,000, 100,000] and an initial value of 40,000.
  - ☐ *pay* getter and setter methods are named `salary()`.
  - ☐ the `toString()` method returns a string in the format

    "*n*\n*x*, *y*\n*r* USD"

    where *n* is *identification*, *x* and *y* are capitalized *lastname* and *firstname*, respectively, and *r* is *pay* with two decimal places.