CMPS 12A – Fall 2017
Programming Assignment 5: Payroll for Startup Company
Due: Friday December 8 @ 11:59pm

Overview
In this assignment you will be implementing 7 total classes that will simulate a payroll of employees of a startup company. The superclass Employee is provided in examples/program5, and the first six classes below will subclass from Employee.

All instance variables are private and thus require necessary getter and setter methods. All instance methods are public.

```
BoardMember extends Employee
     BoardMember(int yearsWorked); // constructor
     final double YEARLY_BONUS = 20000; // $/year
     final int PAID_VACATION = 20; // days
     final int UNPAID_VACATION = 10; // days
     final int SICK_LEAVE = 10; // days
     final double HEALTH_INSURANCE = 20000;//$/year
     final double INCOME = 200000; // $/year

     int usedUnpaidVacation; // keeps track of how much
                             //unpaid vacation has been used.
     int usedVacation; // keeps track of how much paid vacation
                       //  has been used.
     int usedSickDays; // keeps track of how many sick days
                       //  have been used.
     void usePaidVacation(); //  adds one to usedVacation
                             //instance variable.
     void useUnpaidVacation(); // adds one to
                               // usedUnpaidVacation
     void useSickDay(); //  adds one to usedSickDays
     void workYear(); // adds one to yearsWorked.
     double YTDValue(); /* overridden from Employee class.
     YTDValue() is calculated by taking the sum of:
          YEARLY_BONUS
          HEALTH_INSURANCE
          INCOME
          PAID_VACATION * (INCOME / 260)
          (SICK_LEAVE – usedSickDays) * ((INCOME /
          260) / 2) */
     int yearsTillRetirement(); /* overridden from Employee
     class. This should be rounded up to the nearest int. The
     calculation should take place as floating point arithmetic to ensure
     precision (hint: cast as double where necessary). This number
     should not be below zero. This is calculated by: 35 –
```

```
     (yearsWorked + (usedUnpaidVacation / 260) +
     ((usedVacation / 260) * 2) + (usedSickDays /
     260))
     */

Engineer extends Employee
     Engineer(int yearsWorked); // constructor
     final double YEARLY_BONUS = 5000; // $/year
     final int PAID_VACATION = 10; // days)
     final int UNPAID_VACATION = 10; days
     final double HEALTH_INSURANCE = 10000; // $/year
     final double INCOME = 100000; // $/year
     int usedUnpaidVacation; // keeps track of how much
                             //unpaid vacation has been used.
     int usedVacation; // keeps track of how much paid vacation
                       // has been used.
     void usePaidVacation(); // adds one to usedVacation
                             // instance variable.
     void useUnpaidVacation(); // adds one to
                               // usedUnpaidVacation
     void workYear(); // Adds one to yearsWorked.
     double YTDValue(); /* overridden from Employee class.
     This is calculated by taking the sum of:
          YEARLY_BONUS
          HEALTH_INSURANCE
          INCOME
          PAID_VACATION * (INCOME / 260) */
     int yearsTillRetirement(); /* Overridden from
     Employee class. This should be rounded up to the nearest int.
     The calculation should take place as floating point arithmetic to
     ensure precision (hint: cast as double where necessary). This
     number should not be below zero. This is calculated by: 35 –
     (yearsWorked + (usedUnpaidVacation / 260) +
     ((usedVacation / 260) * 2))
     */

Accountant extends Employee
     Accountant(int yearsWorked); // constructor
     final int PAID_VACATION = 10; // days
     final int UNPAID_VACATION = 10; // days
     final double HEALTH_INSURANCE = 15000; // $/year
     final double INCOME = 125000; // $/year
     int usedUnpaidVacation; // keeps track of how much
                             //unpaid vacation has been used.
     int usedVacation; // keeps track of how much paid vacation
                       // has been used.
```

```
    void usePaidVacation(); // adds one to usedVacation
                            // instance variable.
    void useUnpaidVacation(); // adds one to
                              // usedUnpaidVacation.
    void workYear(); // Adds one to yearsWorked.
    double YTDValue(); /* overridden from Employee class.
    This is calculated by taking the sum of:
         HEALTH_INSURANCE
         INCOME
         PAID_VACATION * (INCOME / 260)
         */
    int yearsTillRetirement(); /* Overridden from
    Employee class. This should be rounded up to the nearest int.
    The calculation should take place as floating point arithmetic to
    ensure precision (hint: cast as double where necessary). This
    number should not be below zero. This is calculated by: 35 –
    (yearsWorked + (usedUnpaidVacation / 260) +
    ((usedVacation / 260) * 2))
    */

Custodian extends Employee
    Custodian(int yearsWorked); // constructor
    final int PAID_VACATION = 10; // days
    final int UNPAID_VACATION = 10; // days
    final double HEALTH_INSURANCE = 10000; // $/year
    final double INCOME = 50000; // $/year
    int usedUnpaidVacation; // keeps track of how much
                            //unpaid vacation has been used.
    int usedVacation; // keeps track of how much paid vacation
                      // has been used.
    void usePaidVacation(); // adds one to usedVacation
    void useUnpaidVacation(); // adds one to
                              // usedUnpaidVacation
    void workYear(); // adds one to yearsWorked.
    double YTDValue(); /* overridden from Employee class.
    This is calculated by taking the sum of:
         HEALTH_INSURANCE
         INCOME
         PAID_VACATION * (INCOME / 260)
         */
    int yearsTillRetirement(); /* : must be overridden from
    Employee class. This should be rounded up to the nearest int.
    The calculation should take place as floating point arithmetic to
    ensure precision (hint: cast as double where necessary). This
    number should not be below zero. This is calculated by: 35 –
```

```
      (yearsWorked + (usedUnpaidVacation / 260) +
      ((usedVacation / 260) * 2))
      */
```

```
Marketer extends Employee
      Marketer(int yearsWorked); // constructor
      final double CLIENT_BONUS = 2000; // per new client
      final int PAID_VACATION = 10; // days
      final int UNPAID_VACATION = 10; // days
      final double HEALTH_INSURANCE = 10000; // $/year
      final double INCOME = 100000; // $/year
      int numBonuses; // keeps track of how many commission
                      // based bonuses the employee received.
      int usedUnpaidVacation; // keeps track of how much
                              //unpaid vacation has been used.
      int usedVacation; // keeps track of how much paid vacation
                        // has been used.
      void signDeal(); // adds one to numBonuses.
      void usePaidVacation(); // adds one to usedVacation
      void useUnpaidVacation(); // adds one to
                                // usedUnpaidVacation
      void workYear(); // Adds one to yearsWorked.
      double YTDValue(); /* Overridden from Employee class.
      This is calculated by taking the sum of:
            CLIENT_BONUS * numBonuses
            HEALTH_INSURANCE
            INCOME
            PAID_VACATION * (INCOME / 260)
            */
      int yearsTillRetirement(); // Overridden from
      Employee class. This should be rounded up to the nearest int.
      The calculation should take place as floating point arithmetic to
      ensure precision (hint: cast as double where necessary). This
      number should not be below zero. This is calculated by: 35 –
      (yearsWorked + (usedUnpaidVacation / 260) +
      ((usedVacation / 260) * 2))
      */
```

```
Intern extends Employee
      Intern(int yearsWorked); // constructor
      final int UNPAID_VACATION = 10; // days
      final double HEALTH_INSURANCE = 5000; // $/year
      final double INCOME = 40000; // $/year
      int usedUnpaidVacation; // keeps track of how much
                              //unpaid vacation has been used.
```

```
void useUnpaidVacation(); // adds one to
                          // usedUnpaidVacation
void workYear(); // Adds one to yearsWorked.
double YTDValue(); /* Overridden from Employee class.
This is calculated by taking the sum of:
     HEALTH_INSURANCE
     INCOME
     */
int yearsTillRetirement(); /* Overridden from
Employee class. This should be rounded up to the nearest int.
The calculation should take place as floating point arithmetic to
ensure precision (hint: cast as double where necessary). This is
calculated by: 35 – (yearsWorked + (usedUnpaidVacation
/ 260))
*/
```

`Payroll` – This class will not explicitly subclass from anything. It will have the following instance variables and methods:

```
Payroll(int x); // Constructor
Employee[] employees; // array of x Employee objects.
                      // See constructor.
void sortEmployeesByRetirement(); /* sort employees
in descending order of when the employee will retire. Employees
retiring the soonest will go at the beginning of the array. */
void sortEmployeesByCost(); /* sort employees in
descending order of the YTDValue() of each employee.
Employees having the highest YTDValue() will go at the
beginning of the array. */
```

What to turn in

Submit a zip file of all 7 classes.

Grading Rubric

Each class is worth 2 points. One point will be taken off for the following: improper indentation, lack of comments, unnecessary compiler errors. This program is graded out of 10 points, so any additional points (you can earn up to 15) will count toward your overall programming assignment points.