CSC1016S Assignment 6

Reference Types

Assignment Instructions

This assignment concerns the use of class or 'static' variables and methods, and the general OOP in Java.

Class variables and class methods are those prefixed by the word "static". There is no creation process involved as there is when working with objects. These components simply exist for the duration of program execution.

One use of class variables and methods is for defining collections of constants and routines – much like a Python module. The class "`java.lang.Math`" is an excellent example. It defines constants such as π, and provides routines for calculating square roots, powers etc. We don't create a 'Math' object to make use of these facilities.

In the appendix you will find a set of specifications for the Employee and Shift classes.

On the Amathuba assignment page you will find a ZIP file containing Java code for Week, CalendarTime, Date, Time, Duration and TimeUnit. Employee and Shift classes depend on these classes.

Exercise one concerns implementing the Shift class.
Exercise two concerns implementing the Employee class.
Exercise 3 concerns implementing the Vector class.

Exercise One [30 marks]

Implement the Shift class as specified in the appendix. To evaluate your work, you should use the interactive features of JGrasp and/or construct test code.

The following code fragment demonstrates class behaviour:

```
//
CalendarTime start = new CalendarTime("2/9/2019%22:00");
CalendarTime finish = new CalendarTime("3/9/2019%06:00");
Shift shift = new Shift(start, finish);
System.out.println(shift);

System.out.println(shift.start());
System.out.println(shift.finish());

System.out.println(shift.inWeek(new Week("35/2019")));
System.out.println(shift.inWeek(new Week("36/2019")));
System.out.println(shift.inWeek(new Week("37/2019")));

System.out.println(Duration.format(shift.length(), "minute"));
//
```

The output will be:

```
2/9/2019%22:00:00 - 3/9/2019%06:00:00
2/9/2019%22:00:00
3/9/2019%06:00:00
false
true
false
8 hours
```

Exercise Two [40 marks]

Implement the Employee class as specified in the appendix. To evaluate your work, you should use the interactive features of JGrasp and/or construct test code.

The following (extensive) code fragment demonstrates class behaviour:

```
//
Employee employee = new Employee("Sivuyile Ngesi", "01010125");
System.out.println(employee.name());
System.out.println(employee.UID());
System.out.println(employee.present());
//
System.out.println();
employee.signIn(new Date(1, 9, 2019), new Time(6,00));
System.out.println(employee.present());
employee.signOut(new Date(1, 9, 2019), new Time(18,00));
System.out.println(employee.present());
//
employee.signIn(new Date(2, 9, 2019), new Time(16, 30));
employee.signOut(new Date(3, 9, 2019), new Time(2, 30));
//
employee.signIn(new Date(3, 9, 2019), new Time(18,00));
employee.signOut(new Date(4, 9, 2019), new Time(4,00));
//
System.out.println();
List<Shift> shifts = employee.get(new Week(35, 2019));
for(Shift shift : shifts) { System.out.println(shift); }
//
System.out.println();
System.out.println(Duration.format(employee.hours(new Week(35,
2019)), "minute"));
```

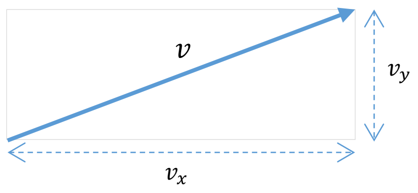
The code produces the following output:

```
Sivuyile Ngesi
01010125
false

true
false

1/9/2019%06:00:00 - 1/9/2019%18:00:00

12 hours
```

NOTE: Each employee has a collection of shifts. You will need some type of collection object. An ArrayList is recommended.

Exercise Three [30 marks]

This exercise concerns the construction of a Java class to represent a vector in two dimensions. A vector has magnitude and direction. The following is a graphical depiction of a vector, v:



A vector can be represented by its horizontal and vertical components i.e. $v = (v_x, v_y)$.

There are a number of operations that can be performed on vectors:

Operation	*Description*
Obtain magnitude	The magnitude of a vector $v = (v_x, v_y)$ is $\sqrt{{v_x}^2 + {v_y}^2}$
Vector add	Given two vectors $v = (v_x, v_y)$ and $v' = (v'_x, v'_y)$, the result of adding them together is the vector $(v_x + v'_x, v_y + v'_y)$.
Scalar Multiply	The multiplication of a vector $v = (v_x, v_y)$ by a value m produces a vector $v' = (mv_x, mv_y)$.
Dot product	Given two vectors $v = (v_x, v_y)$, and $v' = (v'_x, v'_y)$, their dot product is the value of $(v_x \times v'_x, v_y \times v'_y)$.

On the Amathuba page you will find a test harness called *TestVector.java*. Please DO NOT modify this class.

Your task is, given the information above, to develop a Vector class that is compatible with TestVector i.e. TestVector should compile and run without modification.

***Sample I/O (NB:** the input from the user is shown in **bold** – do not program this):*

```
(3) Test scalar multiply(), (4) Test dotProduct()
1
Enter x component and y component (separated by a space):
3 4
Created a Vector object with the given values for vx and vy.
Result of calling getMagnitude(): 5.00

Make a selection and press return:
(0) Quit, (1) Test getMagnitude(), (2) Test add()
(3) Test scalar multiply(), (4) Test dotProduct()
2
Enter x component and y component (separated by a space):
-3 3
Created Vector object: v = (-3.00, 3.00)
Enter x component and y component (separated by a space):
4 -4
Created Vector object: v = (4.00, -4.00)
Result of adding the vectors: v = (1.00, -1.00)
```

```
Make a selection and press return:
(0) Quit, (1) Test getMagnitude(), (2) Test add()
(3) Test scalar multiply(), (4) Test dotProduct()
3
Enter x component and y component (separated by a space):
-3 3
Created Vector object: v = (-3.00, 3.00)
Enter multiplier:
.5
New Vector: v = (-1.50, 1.50)

Make a selection and press return:
(0) Quit, (1) Test getMagnitude(), (2) Test add()
(3) Test scalar multiply(), (4) Test dotProduct()
4
Enter x component and y component (separated by a space):
-3 3
Created Vector object: v = (-3.00, 3.00)
Enter x component and y component (separated by a space):
4 2
Created Vector object: v = (4.00, 2.00)
Result of dot product of the vectors: -6.0

Make a selection and press return:
(0) Quit, (1) Test getMagnitude(), (2) Test add()
(3) Test scalar multiply(), (4) Test dotProduct()
0
```

NOTE: The String class 'format' method can be used to produce a string representation of a real number rounded to 2 decimal places e.g. `String.format("%.2f", 4.896)` produces the result `"4.90"`.

Marking and Submission

Submit your `Shift.java`, `Employee.java`, and `Vector.java` classes all contained within a single .ZIP folder to the automatic marker. The zipped folder should have the following naming convention:

yourstudentnumber.zip

Appendices

Class Employee

An object of this class represents an Employee from the perspective of time keeping. It records the employee name and ID, and logs sign-ins and sign-outs.

Constructors

```
public Employee(String name, String uid)
        // Create an Employee representing the employee with given name and UID.
```

Methods

```
public String name()
        // Obtain this employee's name.

public void UID()
        // Obtain this Employee's ID.

public void signIn(Date d, Time t)
        // Record that this employee has begun a shift on the given date and at the given time.

public void signOut(Date d, Time t)
        // Record that this employee has completed a shift on the given date and at the given time.

public boolean present()
        // Determine whether this employee is present i.e. has signed-in and not yet signed-out.

public List<Shift>  get(Week w)
        // Obtain a list of the shifts worked by this employee during the given week.

public Duration hours(Week w)
        // Returns the total time (hours and minutes) worked during the given week.
```

Class Shift

An object of this class represents a work shift. A shift begins on a given date at a given time and ends on a given date at a given time.

Constructors

public Shift(CalendarTime start, CalendarTime finish)
    // Create a Shift object representing a shift worked between the given date times.

Methods

public static CalendarTime start()
    // Obtain the start date and time for this shift.

public static CalendarTime finish()
    // Obtain the end date and time for this shift.

public static boolean inWeek(Week w)
    // Determine whether this shift occurred within the given week.

public static Duration length()
    // Obtain the length of this shift.

public String toString()
    // Obtain a string representation of this shift of the form "<date>%:<time>-<date>%:<time>".