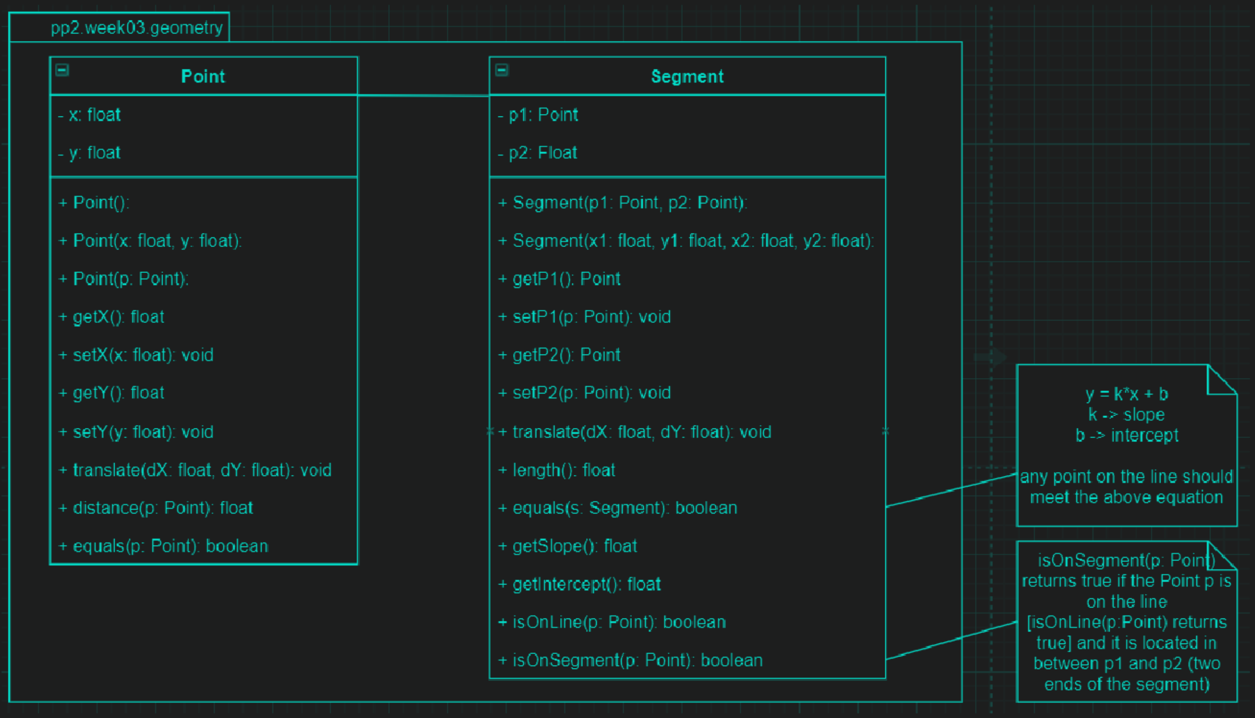
Exercises 3

1. [EX] Given the following UML Class diagram, try to implement it.


pp2.week03.geometry
Point
- x: float
- y: float
+ Point():
+ Point(x: float, y: float):
+ Point(p: Point):
+ getX(): float
+ setX(x: float): void
+ getY(): float
+ setY(y: float): void
+ translate(dX: float, dY: float): void
+ distance(p: Point): float
+ equals(p: Point): boolean
Segment
- p1: Point
- p2: Float
+ Segment(p1: Point, p2: Point):
+ Segment(x1: float, y1: float, x2: float, y2: float):
+ getP1(): Point
+ setP1(p: Point): void
+ getP2(): Point
+ setP2(p: Point): void
+ translate(dX: float, dY: float): void
+ length(): float
+ equals(s: Segment): boolean
+ getSlope(): float
+ getIntercept(): float
+ isOnLine(p: Point): boolean
+ isOnSegment(p: Point): boolean
y = k*x + b
k -> slope
b -> intercept
any point on the line should meet the above equation
isOnSegment(p: Point) returns true if the Point p is on the line [isOnLine(p:Point) returns true] and it is located in between p1 and p2 (two ends of the segment)

   a. Some necessary fixes:
      i. Make sure you have the classes in the appropriate package.
      ii. Complete the Point class, especially add setter/getter methods as well as the last two methods. [Why we needed getter/setters, a.k.a. accessor/modifier?]
      iii. Complete the Segment class.
      iv. From the main methods in the Main class, test each functionality preferably as soon as you complete each.

2. [EX] **Invoice**
   a. Create a class called **Invoice** in package **pp2.week03.ex02** that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as for instance variables
      i. a part number (type String)
      ii. a part description (type String),
      iii. a quantity of the item being purchased (type int)
      iv. a price per item (double).
   b. Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable.
   c. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.
   d. In addition, provide a method named **getInvoiceAmount()** that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value.
   e. Write a test app named **InvoiceTest** that demonstrates class Invoice's functionalities.
   f. Try to draw Class diagram for **Invoice** class.

3. [PW] **CustomDate**
    a. Create a class named CustomDate in package **pp2.week03.datetime** that includes three instance variables—a month (type int), a day (type int) and a year (type int). Provide a constructor that initializes the three instance variables. (Test if the input parameters are valid - consider leap years as well).
    b. Provide a set and a get method for each instance variable.
    c. Provide a method **displayDate()** that displays the month, day and year separated by forward slashes (/).
    d. Provide a method **difference(CustomDate date)** which returns the difference between current method and the input parameter **date** in terms of **days**.
    e. Provide a static method **compare(CustomDate date1, CustomDate date2)** that returns
        i. Positive 1 if the date1 is earlier.
        ii. Negative 1 if the date2 is earlier.
        iii. 0(Zero) if the dates are the same.
    f. Provide a method **displayFormatted()** that will print the date in the format of: **12 Jan 2020**
    g. Write a test app named **CustomDateTest** that demonstrates class **CustomDate's** functionalities.
    h. Try to draw Class diagram for **CustomDate** class.

4. [PW] **CustomTime**
    a. Create a class named **CustomTime** in package **pp2.week03.datetime** that includes three instance variables— hour (type int), minute (type int) and second (type int).
    b. Provide a **constructors**
        i. that will take three input parameters (hour, minute, second respectively) and initialize member variables.
        ii. That will take no input and initialize the members with 0.
        iii. That will take one input parameter (hour) and initialize member variables (minute and second initialized by 0).
        iv. That will take two input parameters (hour, minute, respectively) and initialize member variables (second initialized by 0).
    c. Provide one additional constructor to copy the given CustomTime object and create a new CustomTime object. **public CustomTime(CustomTime time);**
    d. Provide getter methods for each member.
    e. Provide a method **toUniversalString()** which will return a string representing the time in universal format (HH:MM:SS)
    f. Provide a method **toStandardString()** which will return a string representing the time in standard format (H:MM:SS AM/PM)
    g. Write a test app named **CustomTimeTest** that demonstrates class **CustomTime's** capabilities.
    h. Try to draw Class diagram for **CustomTime** class.

5. [EX] Complex numbers
    a. Create a class named ComplexNumber in package **pp2.week03.math** which will be used to represent complex numbers (*x + yi*)
        i. There are (at least) two members of the class: real (x - real part of the complex number) and imag (y - imaginary part of the number both of type float or double.
    b. Provide a constructor to get two real numbers (real and imag) respectively and initialize the members.

c. Provide a method **equals(ComplexNumber number);** that will return true if the object represented by **this** is equal to input parameter **number** object, false otherwise.
d. Provide a method **toString();** to return string representation of the object.
e. Provide a method **re();** that will return the real part of the complex number.
f. Provide a method **imag();** that will return the imaginary part of the complex number.
g. Provide a method **conjugate();** that will return another object of type ComplexNumber which represents the conjugate of **this** complex number object.
   i. conjugate(x+yi) = x – yi
h. Provide a method **abs();** that will return another object of type ComplexNumber which represents the absolute value of **this** complex number object.
i. Provide a method **add(ComplexNumber number);** to add input parameter **number** to the complex number represented by **this** and return the result.
j. Provide a method **sub(ComplexNumber number);** to subtract input parameter **number** from the complex number represented by **this** and return the result.
k. Provide a method **mult(ComplexNumber number);** to multiply input parameter **number** to the complex number represented by **this** and return the result.

l. Write a test app named **ComplexTest** that demonstrates class **ComplexNumber's** capabilities.
   i. **Extra**: Test yourself – Exponentiation: given a complex number **c** and **n** by user find out the value of c^n.
   ii. Input: x, y, n
   iii. Output: $(x+yi)^n$
m. Try to draw Class diagram for **ComplexNumber** class.