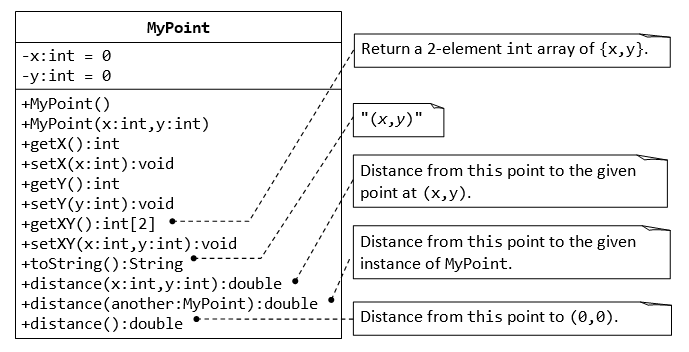
# Assignment 2: (total 170 points)

## Exercise 1: The  MyPoint Classes (60 points)



A class called MyPoint, which models a 2D point with x and y coordinates, is designed as shown in the class diagram. It contains:

* Two instance variables x (int) and y (int).
* A default (or "no-argument" or "no-arg") constructor that construct a point at the default location of (0, 0). (5 points)
* A overloaded constructor that constructs a point with the given x and y coordinates. (5 points)
* Getter and setter for the instance variables x and y.

A method setXY() to set both x and y. (5 points)

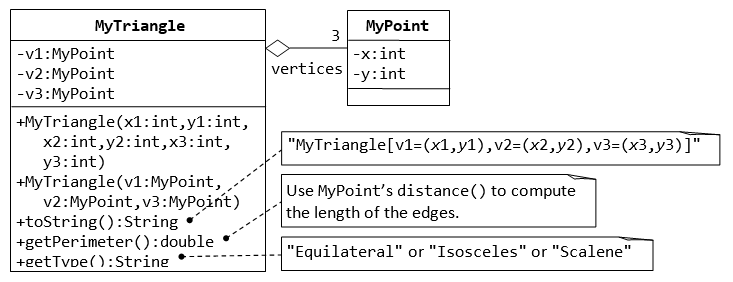
A method getXY() which returns the x and y in a 2-element int array. (5 points)

* A toString() method that returns a string description of the instance in the format "(*x*, *y*)". (5 points)
* A method called distance(int x, int y) that returns the distance from *this* point to another point at the given (x, y) (5 points)
* An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another), e.g., (5 points)
* Another overloaded distance() method that returns the distance from this point to the origin (0,0), e.g., (5 points)

**You are required to:**

* 1. Write the code for the class MyPoint.
  2. Write a test program (called TestMyPoint) to test all the methods defined in the class. (10 points)
  3. Write a program that allocates 10 points in an array of MyPoint, and initializes to (1, 1), (2, 2), ... (10, 10). (10 points)

## Exercise 2: The MyTriangle and MyPoint Classes(35 points)



A class called MyTriangle, which models a triangle with 3 vertices, is designed as shown. The MyTriangle class uses three MyPoint instances (created in the earlier exercise) as its three vertices.

It contains:

* Three private instance variables v1, v2, v3 (instances of MyPoint), for the three vertices.
* A constructor that constructs a MyTriangle with three set of coordinates, v1=(x1, y1), v2=(x2, y2), v3=(x3, y3). (5 points)
* An overloaded constructor that constructs a MyTriangle given three instances of MyPoint. (5 points)
* A toString() method that returns a string description of the instance in the format "MyTriangle[v1=(*x*1,*y*1),v2=(*x*2,*y*2),v3=(*x*3,*y*3)]". (5 points)
* A getPerimeter() method that returns the length of the perimeter in double. You should use the distance() method of MyPoint to compute the perimeter.

(5 points)

* A method printType(), which prints "equilateral" if all the three sides are equal, "isosceles" if any two of the three sides are equal, or "scalene" if the three sides are different. (5 points)

**You are required to:**

1. Write the MyTriangle class.
2. Write a test driver (called TestMyTriangle) to test all the public methods defined in the class. (10 points)

## Exercise 3: MyTriangle and Array (35 points)

Change the three variables of MyPoint type v1,v2,v3 in the MyTriangle class (defined in exercise 2) to an array variable vPoints with 3 array elements of MyPoint type.

**You are required to:**

1. Rewrite the MyTriangle class. Rewrite all methods in the class. (25 points)

Write a test driver (called TestMyTriangle) to test all the public methods defined in the class. (10 points)

## Exercise 4 Inheritance-Implementing the Gourmet Coffee System

**(40 points, each class 10 points)**

### Description

In this assessment, you will implement the classes and relationships illustrated in the following class diagram:

|  |
| --- |
| Figure 1 Portion of Gourmet Coffee System class diagram |
| **Figure 1** *Portion of Gourmet Coffee System class diagram* |

The class specifications are as follows:

#### Class Product

The class Product models a generic product in the store.

Instance variables:

* code. The unique code that identifies the product
* description. A short description of the product
* price. The price of the product

Constructor and methods:

* public Product(String initialCode,
* String initialDescription,
* double initialPrice)

Constructor that initializes the instance variables code, description, and price.

* public String getCode(). Returns the value of instance variable code.
* public String getDescription(). Returns the value of instance variable description.
* public double getPrice(). Returns the value of instance variable price.
* boolean equals(Object object). Overrides the method equals in the class Object. Two Product objects are equal if their codes are equal.
* String toString(). Overrides the method toString in the class Object. Returns the string representation of a Product object. The String returned has the following format:

code\_description\_price

The fields are separated by an underscore ( \_ ). You can assume that the fields themselves do not contain any underscores.

#### Class Coffee

The class Coffee models a coffee product. It extends class Product.

Instance variables:

* origin. The origin of the coffee
* roast. The roast of the coffee
* flavor. The flavor of the coffee
* *aroma*. The aroma of the coffee
* acidity. The acidity of the coffee
* body. The body of the coffee

Constructor and methods:

* public Coffee(String initialCode,
* String initialDescription,
* double initialPrice,
* String initialOrigin,
* String initialRoast,
* String initialFlavor,
* String initialAroma,
* String initialAcidity,
* String initialBody)

Constructor that initializes the instance variables code, description, price, origin, roast, flavor, aroma, acidity, and body.

* public String getOrigin(). Returns the value of instance variable origin.
* public String getRoast(). Returns the value of instance variable roast.
* public String getFlavor(). Returns the value of instance variable flavor.
* public String getAroma(). Returns the value of instance variable aroma.
* public String getAcidity(). Returns the value of instance variable acidity.
* public String getBody(). Returns the value of instance variable body.
* String toString(). Overrides the method toString in the class Object. Returns the string representation of a Coffee object. The String returned has the following format:

code\_description\_price\_origin\_roast\_flavor\_aroma\_acidity\_body

The fields are separated by an underscore ( \_ ). You can assume that the fields themselves do not contain any underscores.

#### Class CoffeeBrewer

Class CoffeeBrewer models a coffee brewer. It extends class Product.

Instance variables:

* model. The model of the coffee brewer
* waterSupply. The water supply (Pour-over or Automatic)
* numberOfCups. The capacity of the coffee brewer

Constructor and methods:

* public CoffeeBrewer(String initialCode,
* String initialDescription,
* double initialPrice,
* String initialModel,
* String initialWaterSupply,
* int initialNumberOfCups)

Constructor that initializes the instance variables code, description, price, model, waterSupply, and numberOfCups.

* public String getModel(). Returns the value of instance variable model.
* public String getWaterSupply(). Returns the value of instance variable waterSupply.
* public int getNumberOfCups(). Returns the value of instance variable numberOfCups.
* String toString(). Overrides the method toString in the class Object. Returns the string representation of a CoffeeBrewer object. The String returned has the following format:

code\_description\_price\_model\_waterSupply\_numberOfCups

The fields are separated by an underscore ( \_ ). You can assume that the fields themselves do not contain any underscores.

#### Class OrderItem

Class OrderItem models an item in an order.

Instance variables:

* product. This instance variable represents the one-way association between OrderItem and Product. It contains a reference to a Product object.
* quantity. The quantity of the product in the order.

Constructor and methods:

* public OrderItem(Product initialProduct,
* int initialQuantity)

Constructor that initializes the instance variables product and quantity.

* public Product getProduct(). Returns the value of the instance variable product, a reference to a Product object.
* public int getQuantity(). Returns the value of the instance variable quantity.
* public void setQuantity(int newQuantity). Sets the instance variable quantity to the value of parameter newQuantity.
* public double getValue(). Returns the product of quantity and price.
* String toString(). Overrides the method toString in the class Object. Returns the string representation of an OrderItem object. The String representation has the following format:

quantity product-code product-price

The fields are separated by a space. You can assume that the fields themselves do not contain any spaces.

#### Test driver classes

Complete implementations of the following test drivers are provided in the student archive. Use these test drivers to verify that your code works correctly.

* Class TestProduct
* Class TestCoffee
* Class TestCoffeeBrewer
* Class TestOrderItem

### Files

The following files are needed to complete this assignment:

* [*student-files.zip*](https://www.icarnegie.com/content/SSD/SSD3/4.2.0.0/normal/pg-class-imp/pg-impl-class/assm-exer-impl-class/pool-pr-impl-class/qn-pr-impl-class-cof-gou-sys/handout/student-files.zip) — Download this file. This archive contains the following:
  + *TestProduct.java*
  + *TestCoffee.java*
  + *TestCoffeeBrewer.java*
  + *TestOrderItem.java*

### Tasks

Implement classes Product, Coffee, CoffeeBrewer, and OrderItem. Document using Javadoc and follow code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. **Extract** the files student-files.zip
2. **Then**, implement class Product from scratch.Use TestProduct driver to test your implementation.
3. **Next**, implement class Coffee from scratch.Use TestCoffee driver to test your implementation.
4. **Then**, implement class CoffeeBrewer from scratch. Use TestCoffeeBrewer driver to test your implementation.
5. **Finally**, implement class OrderItem from scratch. Use TestOrderItem driver to test your implementation.

### Submission

Upon completion, submit **only** the following:

1. Product.java , Product.class
2. Coffee.java , Coffee.class
3. CoffeeBrewer.java , CoffeeBrewer.class
4. OrderItem.java, OrderItem.class