Homework 2

Question 1 (10 pt.)

Create a directory named `q1`. In this directory, create the following classes, where each class is implemented in a file with the same name as the class plus the `.java` extension:

- Class `Point` (2 pt.)

  This is an abstract class representing a point in an *n*-dimensional Euclidean space. It has the following properties:

  - A private string field `name`.
  - A public function `Print()`. It prints "`Point NAME`" into the terminal, where `NAME` is the name of the point.
  - A public function `Read()`. This function takes an object of type `Scanner` as its input argument, which must have been created in the main program. The function prints message "`Enter name:`" and reads the point's name from the keyboard.
  - A public, abstract function `GetNumCoordinates()`, which returns the number of coordinates of the point. Every child of class `Point` must override this function.

- Class `Point2D` (2 pt.)

  This is a child class of `Point`, representing a point in a 2-dimensional coordinate space. It has the following properties:

  - Private fields `x` and `y` of type `double`.
  - A public function `Read()`, which overrides the function with the same name in the parent class. As a consequence, this function has the exact same header as in the parent class.

    The function first reads the point's name from the keyboard by invoking the parent's version of `Read()`. This can be done using statement `super.Read()`, passing the scanner object as an argument.

    The function then prints message "`Enter X:`", and reads a `double` from the keyboard corresponding to the X coordinate, which is saved in field `x`. After reading the `double`, you

need to insert a dummy `scanner.nextLine()` statement in order for Java to ignore the extra newline character introduced when the user presses Enter.

Finally, the function prints message "`Enter Y:`", and reads a double from the keyboard corresponding to the Y coordinate, which is saved in field `y`.

- A public function `Print()`, which overrides the function with the same name in the parent class. This function should invoke the parent's version of `Print()` in order to print the point's name, and continue by printing the X and Y coordinates.

- A public function `GetNumCoordinates()`, overriding the parent's abstract function with the same name, and returning 2.

- A public function `GetDistance()`. This function takes another object of type `Point2D` as an argument, and returns a `double` value containing the Euclidean distance between the point represented by the current instance and the point passed in the argument.

$$distance = \sqrt{(B.x - A.x)^2 + (B.y - A.y)^2}$$

- Class `Point3D` (2 pt.)

  This is a child class of `Point`, representing a point in a 3-dimensional coordinate space. It includes fields `x`, `y`, and `z` of type `double`, plus all other properties present `Point2D` accounting for the fact that there are now 3 coordinates.

- Class `TestPoint2D` (2 pt.)

  This class contains a main program used to test class `Point2D`. The `main()` function starts by instantiating a `Scanner` object, passed later to `Read()` functions.

  The function then instantiates two points `a` and `b` of type `Point2D`, and reads their properties from the keyboard by invoking the `Read()` function on them.

  The function then prints points `a` and `b` by invoking function `Print()` on them.

  The function then prints the number of coordinates in point `a` by invoking function `GetNumCoordinates()` on it. The output should be "`Number of coordinates: 2`".

  Finally, the function prints the distance between points `a` and `b` by invoking function `GetDistance()`. The output should be "`Distance: VALUE`", where `VALUE` represents the distance between the points.

- Class `TestPoint3D` (2 pt.)

  This class contains a separate main program used to test class `Point3D`. The `main()` function carries out the same exact steps as `TestPoint2D`, accounting for the fact that it is now using points in a 3-dimensional space.

Test both main programs in classes `TestPoint2D` and `TestPoint3D`, and verify that they work as expected. Create a package named `q1.zip` containing directory `q1`, and submit it on Canvas.