Homework 3

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. You may start by copying the files created during our lectures and discarding the functionality that is not needed in this question.

- Class `Shape` is an abstract class with the following properties:
  - A private string field called `name`.
  - A public constructor that takes the shape's name as an argument.
  - A public, virtual function `Print()` that prints information about the shape.
  - A public, abstract function `GetArea()` that returns the shape's area.
  - A public function `compareTo()` that compares the current shape with the shape passed in its argument and returns –1, 0, or 1.

- Class `Circle` is a child of class `Shape` with the following properties:
  - A private field of type `double` named `radius`.
  - A public constructor taking the circle's name and radius as arguments.
  - A public function `Print()` printing information about the circle.
  - A public function `GetArea()` returning the circle's area.

- Class `Rectangle` is a child of class `Shape` with the following properties:
  - A pair of private fields of type `double` named `width` and `height`, respectively.
  - A public constructor taking the rectangle's name, width, and height as arguments.
  - A public function `Print()` printing information about the rectangle.
  - A public function `GetArea()` returning the rectangle's area.

a) (3 pt.) Create a new class `Test` containing a private, static function named `Sort()`. This function takes an array of shapes as an argument and sorts it using a selection sort algorithm. You may start by copying the selection sort algorithm implemented in class, and adapting it to the fact that the array elements are now of type `Shape` instead of `int`. This affects function `Sort()` itself, as well as functions `GetMinIndex()` and `Swap()`, which you need to include in the class. You are no longer able to rely on relational operators (`<`, `>`, `==`) to compare elements; instead, you need to invoke the `compareTo()` function on the shapes for this purpose.

b) (7 pt.) Write a `main()` function in class `Test` performing the following actions:

- Ask the user for a number of shapes *N*.
- Allocate an array of *N* shapes, where each shape is initially a *null* reference.
- Repeat the following actions *N* times:
  - Ask the user for a shape type (`'Circle'` or `'Rectangle'`).
  - Read the shape's name from the user.
  - Read other shape properties depending on the type of shape (radius vs. width and height).
  - Instantiate the specific shape and save it at the appropriate position of the array.
- Sort the array of shapes by invoking the `Sort()` function implemented before.
- Print the array of shapes by invoking function `Print()` on every element of the array. Here you can observe the power of polymorphism: in an array of objects of type `Shape`, each invocation to `Print()` has a different effect based on the actual type of instance (`Circle` or `Rectangle`) on top of which it is being invoked.

Run your main program by entering various shapes of different types, and verify its correct execution. Create a package named `q1.zip` containing directory `q1`, and submit it on Canvas.