# Assignment 3

Object Orientation

Spring 2020

## 1 Learning Objectives

After making this exercise you should be able to

- Define new interfaces
- Make classes that implement one or more interfaces
- Use interfaces instead of classes as method argument and array elements
- Sort arrays of objects using `compareTo<T>` en `Comparable<T>`
- Recognize more complex input with a `Scanner`

## 2 Geometric Objects

Geometric objects like circles and rectangles have some different attributes, but also some similar attributes and methods. An interface is a convenient way to model the common operations on geometric objects in Java. In this way we can use geometric objects, without bothering whether they are circles or rectangles.

For the geometric objects in this exercise the common operations are:

- Four methods that return the left-, right-, bottom- and top-border of the smallest surrounding rectangle as a `double`. For example, a circle that has its centre at the coordinates (1,2) and radius 1, the left-, right-, bottom- and top-border will be 0, 2, 1 and 3 respectively.

- A method that yields the area of the object as a `double`.

- A method to move the object over the given distances $dx$ and $dy$, both of type `double`.

- The method `compareTo` from the Java `Comparable<T>` interface that compares the area of geometric objects[1].

All these methods should be part of an interface, and circles and rectangles should implement the interface. A circle is defined by the position of its centre and its radius. A rectangle is characterized by its lower left corner, its width and height. Each of these objects has a tailor-made `toString` method showing its characteristics.

Your interactive program should execute user commands which can create and manipulate geometric objects. Therefore, your program needs to keep track of an array of geometric objects. The array should have a fixed size, for example 10. After each command the system lists the current geometric objects.

The commands to be recognized are:

[1] See `docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`.

**quit** - stops the program.

**show** - lists the geometric objects.

**circle** $x\ y\ r$ - adds a circle at $(x,y)$ with radius $r$.

**rectangle** $x\ y\ w\ h$ - adds a rectangle with $(x,y)$ for the lower left corner, with width $w$ and height $h$.

**move** $i\ dx\ dy$ - moves object $i$ over the specified distance in $x$ and $y$ direction.

**remove** $i$ - deletes object $i$.

**sort** $c$ - sorts the objects in the array according to criterion $c$. The argument $c$ is optional.

- When $c$ is ”x”, objects should be sorted by their left most point from small to large.
- When $c$ is ”y”, objects should be sorted by their bottom point from small to large.
- Without argument the objects are sorted on their area from small to large.

As always you should make a clean separation in your code between input-output handling and the actual logic.

## 3 Sorting Arrays in Java

Sorting arrays is used very often in programs. The Java API contains reusable and efficient solutions. The class `Arrays`[2] provides the static sorting methods

```
sort(Object[] a)
sort(Object[] a, int fromIndex, int toIndex)
```

These methods sort the elements according to the ordering of the objects using the `Comparable<T>` interface[3]. Use this to sort an array `a` as `Arrays.sort(a)`. `Comparable` contains only the method `int compareTo(T o)`. Your geometrical object interface should extend this interface, so that the classes rectangle and circle provide an actual implementation for the `int compareTo(T o)` method.

In this method, `T` is the type of object to be compared. In Java it is called a *generic* argument, which will be covered later in this course. For now replace `T` with the name of the class or interface you want to use as argument in `Comparable<T>`. For example, when you want to compare objects that implement the `Geometric` interface you have to implement `int compareTo(Geometric o)`. When a class `C` should implement `Comparable`, you write `public class C implements Comparable<C>`.

When you implement `o1.compareTo(o2)`, it should return `0` when the objects are equal. It should return a negative number when `o1` is smaller than `o2`. Otherwise, it should return a positive number. Implement this interface to compare geometric objects based on their area.

[2]See `docs.oracle.com/javase/8/docs/api/java/util/Arrays.html`.
[3]See `docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`.

For the other sorting criteria, you cannot use the `Comparable<T>` interface again since a class can implement an interface only once. The class `Arrays` offers static sorting methods that use an additional object which implements the interface `Comparator` [4] This interface provides the method `int compare(T o1, T o2)`. The integer result obeys the same rules as the `Comparable<T>` interface.

```
sort(T [] a, Comparator<T> c)
sort(T [] a, int fromIndex, int toIndex, Comparator<T> c)
```

These sorting methods take a comparator `c` which is used to compare the objects in the array `a`.

## 4 Your Tasks

- Create an interface called `Geometric` for geometric objects and classes `Circle` and `Rectangle` implementing this interface.
- Ensure that all geometric objects can be sorted by implementing the `Comparable <T>` interface.
- Introduce other classes as needed to achieve a well designed program that separates I/O and logic.

The classes `Arrays`, `Comparable` and `Comparator` are part of the Java standard library. You have to use them. It might be helpful to draw a class diagram before you start coding, but it is not required to hand in this diagram. It is convenient to construct the parts of this diagram incrementally and to test the finished parts as early as possible, before your entire program in completed.

## 5 Submit Your Project

To submit your project, follow these steps.

1. Use the **project export** feature of NetBeans to create a zip file of your entire project: File → Export Project → To ZIP.
2. **Submit this zip file on Brightspace**. Do not submit individual Java files. Do not submit any other archive format. Only one person in your group has to submit it. Submit your project before the deadline, which can be found on Brightspace.

[4] See `docs.oracle.com/javase/8/docs/api/java/util/Comparator.html`