Software Architecture. Interfaces exercise:

1. Implement a class Person that: a. Stores both name and surname. b. Has a method to print the name and surname to the screen.

```
public class Person {
    private String firstName;
    private String surname;

    public Person(String firstName, String surname){
        this.firstName = firstName;
        this.surname = surname;
    }

    @Override
    public String toString() {
        return (this.firstName + " " + this.surname);
    }
    public String getFirstName(){ return this.firstName;}
    public String getSurname(){return this.surname;}
}
```

2. Implement a class Sorter that it is able to sort an array of Persons (Person[]). It must: a. Have a method that receives an array of Person b. Sort the array by surname, name. c. Use a simple algorithm like: i. https://en.wikipedia.org/wiki/Bubble_sort ii. https://en.wikipedia.org/wiki/Insertion_sort

```
import java.util.List;
import java.util.ArrayList;

public class Sorter {
    private List<Person> people = new ArrayList<>();

    public void addPerson(String firstName, String surname){
        people.add(new Person(firstName, surname));
    }

    public List<Person> getPeople(){
        return people;
    }

    public void sortPeople(){
        int index_i = 0;
        int index_j;

        while (index_i < people.size()){
            index_j = index_i;
            while (index_j > 0) {
                Person prev = people.get(index_j-1);
                Person curr = people.get(index_j);

                int cmp = prev.getSurname().compareTo(curr.getSurname());
                if (cmp == 0){
                    cmp = prev.getFirstName().compareTo(curr.getFirstName());
                }
                if (cmp > 0){
                    people.set(index_j - 1, curr);
                    people.set(index_j, prev);
                }
                index_j--;
            }
            index_i++;
        }

    }
}
```

3. Implement a Program that uses the Sorter to sort an array of 5 persons.

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        Sorter sorter = new Sorter();
        sorter.addPerson("henry","Blue");
        sorter.addPerson("anna", "Blue");
        sorter.addPerson("billy","Joe");
        sorter.addPerson("susan", "Smith");
        sorter.addPerson("john", "Adams");
        sorter.addPerson("alice", "Cooper");

        List<Person> personList = sorter.getPeople();
        for (Person person : personList) {
            System.err.println(person);
        }
        sorter.sortPeople();
        System.err.println("");

        personList = sorter.getPeople();
        for (Person person : personList) {
            System.err.println(person);
        }
    }
}
```

output:

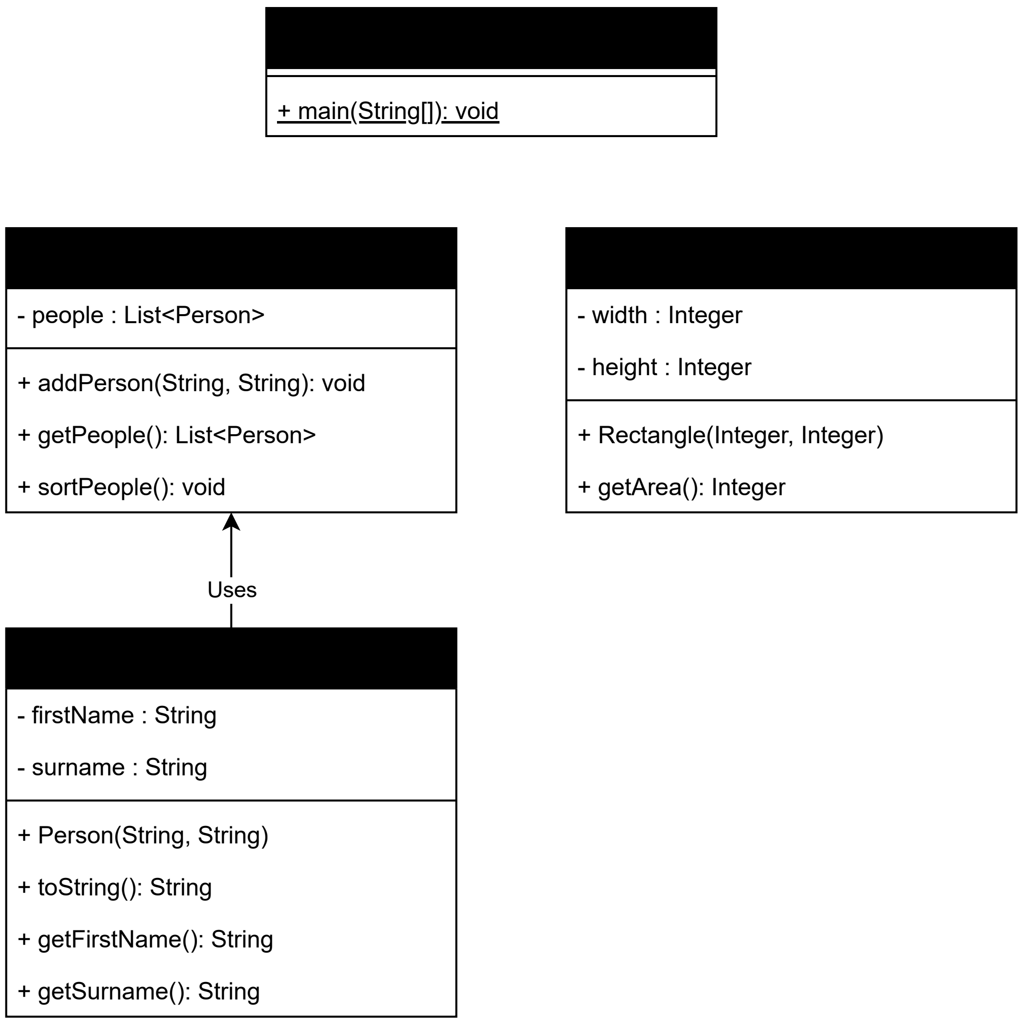
```
anna Blue
billy Joe
susan Smith
john Adams
alice Cooper

john Adams
anna Blue
henry Blue
alice Cooper
billy Joe
susan Smith

```

4. Implement a class Rectangle that: a. Stores its width and the height b. Has a method to calculate its area

```java


public class Rectangle {
    private Integer width;
    private Integer height;

    public Rectangle(Integer width, Integer height){
        this.width = width;
        this.height = height;
    }

    public Integer getArea(){
        return this.width * this.height;
    }

}
```

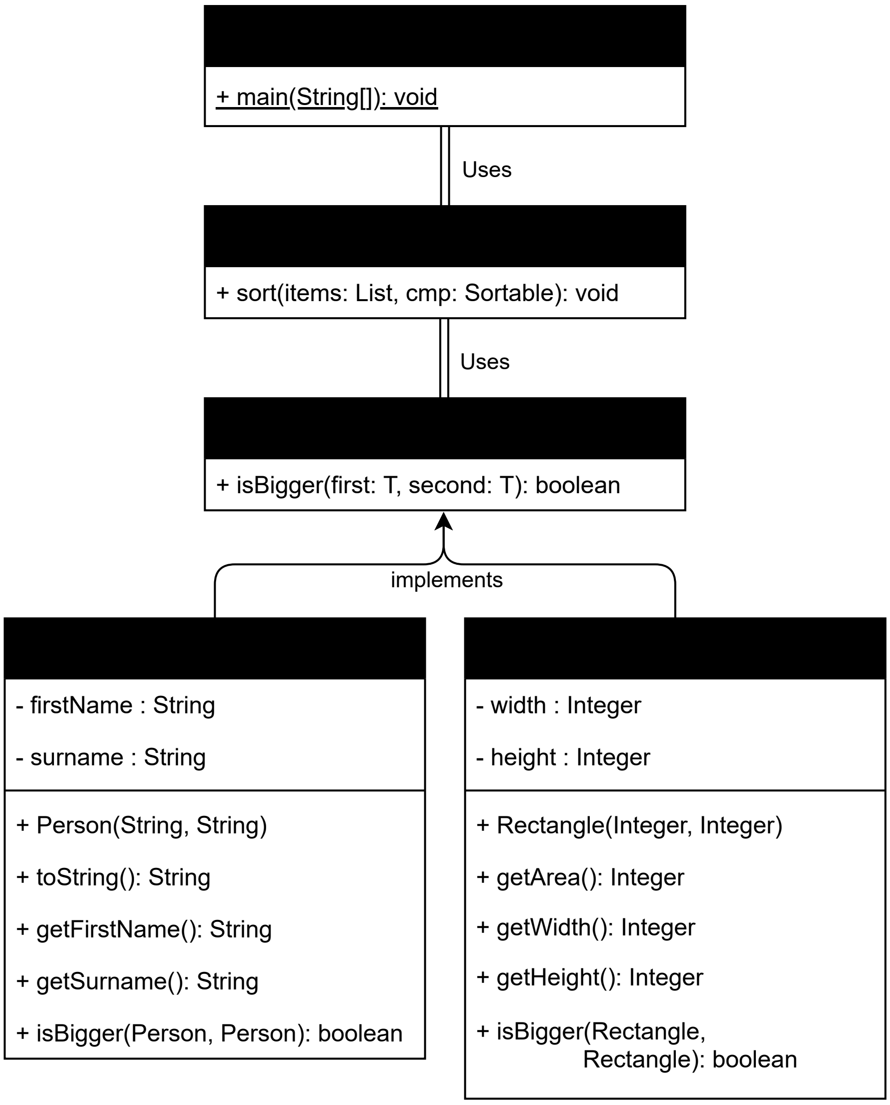
5. Draw a diagram of the current classes of the system

Bellow is a simple representation of the classes and their relationships:



6. Which modifications you have to do to make the Sorter class capable of sorting both Rectangles and Persons? a. Rectangles must be sorted by its area. b. Which element of your design enables the class Sorter and its sort method to accept both Persons and Rectangles? c. Draw a new diagram with the proposed changes before trying to implement it. d. DO NOT CONTINUE READING UNTIL YOU HAVE A CLEAR IDEA OF HOW TO IMPLEMENT IT !!

To make the Sorter class capable of sorting both Person and Rectangle, I made a generic interface Sortable with a method isBigger(T first, T second) that defines the ordering rule. Both Person and Rectangle implement this interface: Person compares by surname then firstName, and Rectangle compares by area. The Sorter class was then remade to a generic sort method sort(List, Sortable), which uses the isBigger method, allowing it to sort single type lists of either Person or Rectangle. The updated class diagram bellow shows how Sorter depends on the Sortable interface, and how both Person and Rectangle implement it.



7. This element must have a method that receives two objects and return a boolean indicating if the second is bigger or not (in sort order) than the the first object. Make sure that your design follows this tip.

this is done with the above explanation 😃

8. Implement your design and make a program that sorts an array of 5 Persons and other array of 5 Rectangles.

You can see the final code in the repository ShapesAndPeople here:

https://github.com/henryblu/SOFTENG/tree/main/ShapesAndPeople/src

With this, we update main to the following:

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {

        List<Person> people = new ArrayList<>();
        people.add(new Person("henry","Blue"));
        people.add(new Person("anna", "Blue"));
        people.add(new Person("billy","Joe"));
        people.add(new Person("susan", "Smith"));
        people.add(new Person("john",  "Adams"));
        people.add(new Person("alice", "Cooper"));

        System.out.println("-- People BEFORE --");
        for (Person p : people) System.out.println(p.getFullName());

        Sorter.sort(people, new Person("", ""));
        System.out.println("-- People AFTER  --");
        for (Person p : people) System.out.println(p.getFullName());


        List<Rectangle> rects = new ArrayList<>();
        rects.add(new Rectangle(4,5));
        rects.add(new Rectangle(2,12));
        rects.add(new Rectangle(3,3));
        rects.add(new Rectangle(10,1));
        rects.add(new Rectangle(6,4));

        System.out.println("-- Rect areas BEFORE --");
        for (Rectangle r : rects) System.out.println(r.getArea());

        Sorter.sort(rects, new Rectangle(1,1));
        System.out.println("-- Rect areas AFTER  --");
        for (Rectangle r : rects) System.out.println(r.getArea());
    }
}
```

which has the following output:

```
-- People BEFORE --
henry Blue
anna Blue
billy Joe
susan Smith
john Adams
alice Cooper
-- People AFTER  --
john Adams
anna Blue
henry Blue
alice Cooper
billy Joe
susan Smith
-- Rect areas BEFORE --
20
24
9
10
24
```

```
-- Rect areas AFTER  --
9
10
20
24
24
```

9. Your solution should be similar to the one, already present in standard Java. Have a look at the following documents: a. https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html b. https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort(java.util.List)

This is also done with the above code 😃

10. Implement your program to sort Persons and Rectangles by using the Comparable interface and the Collections class.

To implement the same functionality using Java's Comparable interface and Collections class, we need to make both Person and Rectangle implement Comparable. This was done and demonstrated in the /srcComparable directory of the repository. Found here:

https://github.com/henryblu/SOFTENG/tree/main/ShapesAndPeople/srcComparable