Design patterns lab 2

Reference:
Read descriptions for **Strategy, Adapter, Chain of Responsibility** in the **Lecture Notes**
Additional lecture slides
https://www.oodesign.com/
https://medium.com/@ronnieschaniel/object-oriented-design-patterns-explained-using-practical-examples-84807445b092
https://en.wikipedia.org/wiki/Chain-of-responsibility_pattern

Exercise 1
In Java framework **Strategy** pattern is used in Collection framework for sorting. Eg. sort method has a default Comparator which can be replaced by own implementing classes. In this execise you will reimplement this approach without using classes or interfaces from the collection class.
In more detail: MyCollections class has an overloaded sort method, one that receives an array of generic comparable objects, one that receives an array of array of generic objects and a MyComparator. Implement the classes and test them using Junit tests. Signatures for the classes and methods are given below.

```
public class MyCollections {
      public static <T extends Comparable<T>> void sort(T[] arr){
      …
      }
      public static <T> void sort(T[] arr,MyComparator<T> comp){
            …
}
public interface MyComparator<T> {
      public int compare(T t1, T t2);
}
public class IntegerAscendingComparator implements MyComparator<Integer> {…}
```

End of exercise 1

Exercise 2
Adapter pattern is used to adapt interfaces to clients using different interfaces, sometimes also transforming the data into appropriate forms.
For this exercise implement a class PersonalData with the interface PersonalDataI with the methods getName():String, getBDay():LocalDate, getEmail():String, getTelephone():String.
However the client uses the interface PersonalInformation with the method getPersonalInformation():String that returns a JSON with the fields name, year of birth, email and telephone.
Use Adapter pattern to adapt the new interface to the existing interface. Implement needed classes and interfaces and test the methods in Junit tests. Write the design class diagram for the exercise, following Adapter design pattern structure.
End of exercise 2

Exercise 3
We will use Chain of Responsibility pattern for this exercise. A sensor continuously (use a Thread) produces random SensorEvents. Each SensorEvent has a type (Fire, Intrusion, Water, Temperature), a timestamp and a location (the room in which the event was produced).
The event is dispatched to a chain of NotificationServices:
EmailNotification handles all Fire, Intrusion, Water events, it prints on the console “Email was sent for event (details)”
TelephoneNotification handles only Fire and Intrusion events, it prints on the console “A call was made for event (details)”
Logger handles all events, it prints on the console “Event (details) was logged”.
Implement all classes and interfaces, draw the design class diagram, and test the program with a main method that sets up the chain of handlers and continuously dispatches random events to the chain.
End of exercise 3