Problem 1

Write a class-comparator **MyComp**, objects of which can be used for sorting an array of **Integer**'s according to various criteria. Desired criterium can be selected by a value of an integer field initialized in the constructor: possible values of the field should correspond to the values of static final constants of type **int** defined in the class:

- BY_VAL: by numerical value, in ascending order;
- BY_VAL_REV: by numerical value, in descending order;
- BY_NUM_OF_DIVS: by number of divisors;
- BY_SUM_OF_DIGS: by sum of digits.

If two element are equal according to a selected criterium, the numerical values decide (in ascending order). For example, the following program

download *Compars.java*

```
import java.util.Arrays;
import java.util.Comparator;

class MyComp implements Comparator<Integer> {
    public static final int
                BY_VAL=0, BY_VAL_REV=1,
                BY_NUM_OF_DIVS=2, BY_SUM_OF_DIGS=3;
    // ...
}

public class Compars {
    public static void main(String[] args) {
        Integer[] a = {1,5,33,12,98,15};
        printTable("Original    ", a);

        Arrays.sort(a,new MyComp(MyComp.BY_VAL));
        printTable("ByVal       ", a);

        Arrays.sort(a,new MyComp(MyComp.BY_VAL_REV));
        printTable("ByValRev    ", a);

        Arrays.sort(a,new MyComp(MyComp.BY_NUM_OF_DIVS));
        printTable("ByNumOfDivs ", a);

        Arrays.sort(a,new MyComp(MyComp.BY_SUM_OF_DIGS));
        printTable("BySumOfDigs ", a);
    }
```

```
        static void printTable(String mess, Integer[] a) {
            System.out.print(mess + "[ ");
            for (int d : a) System.out.print(d + " ");
            System.out.print("]\n");
        }
    }
```

should print something like

```
Original    [ 1 5 33 12 98 15 ]
ByVal       [ 1 5 12 15 33 98 ]
ByValRev    [ 98 33 15 12 5 1 ]
ByNumOfDivs [ 1 5 15 33 12 98 ]
BySumOfDigs [ 1 12 5 15 33 98 ]
```

[Note: you can use enums instead of final static fields, the values of which determine the comparisons criteria]

Problem 2

Define an interface **Reversible** with one method

```
    Reversible reverse();
```

and implement it in classes **ReversibleString** and **ReversibleDouble**. The method **reverse** reverses the order of characters for **String**s and calculates the reciprocal for numbers (i.e., for string 'cat' we should get 'tac' and for number 3 the result is 0.3333). The method returns *this* **Reversible** with current value (reversed).

The following program

download *SReversible.java*

```
    public class ReversibleTest {
        public static void main(String[] args) {
            Reversible[] revers = new Reversible[] {
                    new ReversibleString("Cat"),
                    new ReversibleDouble(2),
                    new ReversibleDouble(3),
                    new ReversibleString("Dog"),
                    new ReversibleString("Alice in Wonderland"),
                    new ReversibleDouble(10),
            };

            System.out.println("Original:");
            for (Reversible r : revers) System.out.println(r);

            for (Reversible r : revers) { r.reverse(); }

            System.out.println("Reversed:");
            for (Reversible r : revers) System.out.println(r);

            System.out.println("Reversed again and modified:");
            for (Reversible r : revers) {
                /*<- ... */
            }
        }
    }
```

should print

```
    Original:
    Cat
    2.0
    3.0
    Dog
    Alice in Wonderland
    10.0
    Reversed:
    taC
    0.5
    0.3333333333333333
    goD
    dnalrednoW ni ecilA
    0.1
    Reversed again and modified:
    Text: Cat
    12.0
    13.0
    Text: Dog
    Text: Alice in Wonderland
    20.0
```

Note: in the last loop all strings are prepended by the word `Text:`, while numbers are incremented by 10.
Note: the code of **ReversibleTest** class may (and should) be modified only in the place marked with `/*<- ... */`.

Problem 3

Define a (functional) interface **SFilter** declaring one method **test** which takes a **String** and returns a **boolean**. The interface defines also a static function

```
    public static String[] filter(String[] arr,
                                  SFilter filt) {
        // ...
    }
```

which takes an array of **String**s and an object implementing the **SFilter** interface. The function returns another array of **String**s which contains only those elements of the original array for which calling **test** on filt returns **true**.
Define also a class **LenFilter** which implements the interface. The class has one field minLen of type **int** which is set by the constructor; the implementation of the function **test** returns **true** if, and only if, the string passed as the arguments is of length at least minLen.
In the **main** function create an array of **String**s and then call the **SFilter.filter** function passing this array and implementation of the **SFilter** interface in the form of:

- an object of **LenFilter** initialized in this way that its **test** method selects only strings of the length greater than 4;
- an object of an anonymous class which implements the **SFilter** interface in such a way that it selects only strings whose first letter is earlier in the alphabet that 'D' but later or equal 'A';
- a lambda which selects only those strings whose first letter is later in the alphabet than 'H' but earlier or equal 'Z'.

The following program:

download *StringFilter.java*

```java
import java.util.Arrays;

@FunctionalInterface
interface SFilter {
    // ...
}

class LenFilter implements SFilter {
    // ...
}

public class StringFilter {
    public static void main (String[] args) {
        String[] arr = {"Alice", "Sue", "Janet", "Bea"};
        System.out.println(Arrays.toString(arr));

        String[] a1 = SFilter.filter(arr, /* ... */);
        System.out.println(Arrays.toString(a1));

        String[] a2 = SFilter.filter(arr, /* ... */);
        System.out.println(Arrays.toString(a2));

        String[] a3 = SFilter.filter(arr, /* ... */);
        System.out.println(Arrays.toString(a3));
    }
}
```

should print

```
    [Alice, Sue, Janet, Bea]
    [Alice, Janet]
    [Alice, Bea]
    [Sue, Janet]
```