[Write on your own sheets of paper. On each sheet, on the **front**, write in the **top right corner** your **lastname, firstname(s), group id and (A)** . Suggested time: 40 minutes]

1. [2p] Write Java code for starting a thread implementing the **`Runnable`** interface.

2. [2p] What will be printed when the following java code fragment is executed. Motivate your answer.

```
public class ClassX implements Multiplier{
    int val;
    public ClassX(int val){
            this.val = val;
            System.out.println("Ctor ClassX: " + val);
       }
    public int changeIt (int val) {
        return val * 3;
    }
}
```

```
public class ClassT{
    protected static double SEVEN = 8;
    public ClassT (double two) {
        System.out.println("Ctor ClassT: " + SEVEN);
        SEVEN = two;
    }
}
```

```
 public interface Multiplier {
    int changeIt(int val);
}
```

```
public class ClassA extends ClassT implements Multiplier{
    public static final int ONE = 1;
    public static float TWO = 2;
    public ClassA() {
            super(ONE);
            System.out.println("Ctor ClassA(): " + ONE + " " + SEVEN); }
    public float doIt(int i) {
            return TWO * i;
       }
    public int changeIt (int val) throws UnsupportedOperationException {
        throw new UnsupportedOperationException("Not implemented");
    }
    public static void main(String[] args){
        Object o1 = new ClassA();
        Object o2 = new ClassA();
        ClassX b = new ClassX((int)TWO);
        SEVEN = ((ClassA)o1).doIt(-1);
            System.out.println("ClassA: " +
               ((ClassA)o2).doIt(7) + " " + ((ClassA)o1).SEVEN + b.changeIt(1));
        int val = ((ClassA)o2).changeIt(2);
        System.out.println("Done");
    }
}
```

3. [1p] Draw a class diagram, showing all variables and methods, for the code of exercise 1.

4. [1p] What is the **`throws`** clause used for and what should one put in such a clause?

5. [1p] What does *functional testing* mean?

6. [1p] List the benefits of logging.

7. [1p] What are the main differences between classes **`Vector`** and **`ArrayList`**?

Wandering whirlpools

On a river there are floating whirlpools coming down the stream, ships, and a bridge. All what happens on the river should be shown on the console. It is a *process carried out in steps*, described below:

1. **Whirlpools** -- depicted by **`'W'`** letters if they are in the visible river area (see Fig. 1 below):
   - The initial number of whirlpools is read from the initial configuration, from **`System.in`**.
   - All the whirlpools are initially at the top -- i.e. no whirlpool in the water.
   - The number of whirlpools supplied for a column is depicted by a single decimal digit.
   - On every step:
     - Every whirlpool moves downwards with one position in one of the directions: SW, S or SE, at random
     - One more whirlpool enters the visible water area from the ones at the top.
     - If a whirlpool gets on a position with a ship, the ship is swallowed by the whirlpool and sinks, and both the whirlpool and the ship disappear.
2. The **bridge** -- depicted by **`'B'`** and **`'='`** symbols
   - It is located at the bottom (last line).
   - It is damaged when a ship hits one of its pillars and the game ends, with the message '**`Bridge was hit`**'.
3. The **river** -- symbolized with dots (`'.'`). It flows downwards.
4. **Ships** -- symbolized by **`'S'`** letters:
   - There are ships in the water, placed initially as specified in the start configuration.
   - On every step:
     - If not in the column of a whirlpool, every ship moves one position downwards (i.e. south), if no other ship is in that position; otherwise the ship does not move .
5. **Steps**
   - A step is taken when the user presses the 'Enter' key. First, whirlpools are moved, and then the ships.
   - The process ends when there are no more whirlpools -- to release and in the water, or the bridge was damaged by a ship hitting a pillar.
   - At the beginning of the game, and also after each step you should print to **`System.out`** – *text mode* – the current configuration – see Fig. 1 below.

The start configuration (see example of Fig. 1), contains:

- one line of representing the whirlpools supply. For the example of Figure 1, there are 5 whirlpools in column 10, 4 whirlpools in column 17, 3 in column 25, and so on
- 14 lines of river water, possibly with floating ships, and
- one line of bridge.

The start configuration is read from **`System.in`**. The class **`oop.Exam`** provides the static methods **`readWater()`**, **`readWhirlpools()`**,and **`readBridge()`**. The methods return data read from `System.in` as follows:

- **`readWhirlpools()`** – a 2D array of **`int`** having on row 0 the column of the whirlpools supply and in row 1, at the same index, the number of whirlpools.
- **`readWater()`** – a 2D array of **`int`** having on row 0 the actual row of a ship, and in row 1 the column of that ship, at the same index.
- **`readBridge()`** – a 1D array of integer indexes for the bridge pillar positions

Draw the class diagram and develop a java standalone program to simulate this game. Don't forget to briefly document it.

```
5(10) 4(17) 3(25) 1(36) 9(49) 3(66) 4(75)
.....................................................................................
.....................................................................................
..........................................................................S..........
..........................................................S..........................
..................S....................S.............................................
.....................................................................................
.....................................................................................
.........S...........................................................................
.....................................................................................
..........................S.....................S......................S.............
.....................................................................................
......................................S..............................................
.....................................................................................
.....................................................................................
=========B==========B==========B==========B==========B==========B==========B=========
Press Enter....
```

Fig. 1. An example of a start configuration.