[Write on your own sheets of paper. On each sheet, on the front, write in the **top right corner** your **family name, first name(s), group id and (K)** . Leave a 2cm margin at the left on the front page / at the right on the back for stapling. Closed book. Time: 35min]

1. [2p] What will be printed when the following java code fragment is executed. Motivate your answer.

```
public class A extends B{
    public static final int ONE = 1;
    public static float THREE = 2;
    A ()
    {
        super(ONE);
    }
    public float doIt(int i) {
        return THREE * i;
    }
    public static void main(String[] args){
        Object o1 = new A();
        Object o2 = new A();
        THREE = ((A)o1).doIt(3);
            System.out.println(((A)o2).doIt(4) + " " + ((A)o1).THREE);
    }
}
public class B{
    public static double THREE = 2;

    public B (double two)
    {
        System.out.println("B 3 = " + THREE);
        THREE = two;
    }
}
```

2. [1p] Draw the class diagram for the code of item 1.

3. [3p] Write a **Node** class for a singly-linked list holding integer values, which includes a method for insert in ascending order. Make it from scratch (do not use the List collection type). Also provide a constructor with arguments for nodes.

4. [1p] How can one store primitive types in an `ArrayList` collection?

5. [1p] What is a `ByteBuffer` used for? Give a brief example.

6. [1p] What is the keyword `this` used for in Java?

[Write on your own sheets of paper. On each sheet, on the front, write in the **top right corner** your **family name, first name(s), group id and** (**K**) . Leave a 2cm margin at the left on the front page / at the right on the back for stapling. Open book. Time: 100 minutes]

Close to the North Pole

You are to develop a standalone application which simulates a hunt. The simulation is viewed on a rectangular board. Its dimensions, given on the command line, are bound as follows: 10 ≤ rows ≤ 12 and 20 ≤ columns ≤ 22. The board represents a floating ice area in the northern part of the world. In that area there is one bear, a number of seal holes, between 2 and 6, and a number of seals – between 3 and 7. This numbers are read from the configuration file, named `Bear_Seals.txt` when the game starts. To help you acquire these values, there is a class called `Position`, included in the package `ro.utcluj.cs.northhunt`, which contains a number of useful static methods:

- `int[] getBearPos(String line)` – if given a String with a bear position returns an array with 2 numbers: row and column in positions 0 and 1 of the array
- `int[] getMany(char identifier, String line)` – if given a String with hole positions and first argument 'H' or a String with seals positions and first argument 'S' – returns an array with rows and columns of holes/seals: at even numbered indexes — rows, at odd numbered indexes – corresponding columns
- Use the above static methods to process the line read from the input.
- A seal can swim under water for up to 5 steps. After taking a breath the seal moves away at some random position under the ice, leaving the hole empty for another seal. At some moment (from 1 and up to 5 steps away from the previous breath) some seals will pop up to the nearest hole to take a fresh breath of air. If they get to a hole where the bear is waiting, the first to reach they gets eaten. A seal can die if it does not get to a hole for more than 5 steps.
- The bear, by capital letter 'B', waits at one of the seal holes for a seal to come and take a fresh breath. The number of steps the bear waits for a seal to appear at a hole is between 2 and 6. After eating one seal, the bear will not move or eat for the next 3 steps. The moves of the bear are random, i.e. after the wait period, if no seal appears, the bear will jump to a *randomly chosen* hole. The bear is initially upon a hole (actually in very close proximity) waiting for a seal. That means that there is a hidden hole where the bear waits from the very beginning.
- A step is taken when the user presses the 'Enter' key. At any step, bears move first.

Figure 1 shows an example initial configuration and how it is represented in the text file. At the beginning of the game, and also after each step, you should print to `System.out` – *text mode* – the current configuration. On that printout, the bear is a 'B', the seals beneath ice are lowercase 's's, the holes with seals are 'S's, the holes with no seals with 'H's, and the rest of the squares with dots ('.').
The game ends when the bear has eaten all seals or the seals have all died or 300 steps passed without this unhappy (for the seals, of course) event to occur.

For the following contents of `Bear_Seals.txt`:	the initial printout is:

`B 5 11`
`H 7 3 H 8 18 H 2 16 H 2 18 H 1 19`
`S 1 3 S 8 18 S 3 5 S 0 0 S 1 1 S 0 1 S 9 19`
	`  00000000001111111111`
	`  01234567890123456789`
	`0ss.................H`
	`1.s.s...............H`
	`2................H.H.`
	`3.....s.............`
	`4....................`
	`5...........B........`
	`6....................`
	`7...H..............s.`
	`8.................S.`
	`9...................s`
	`Press Enter...`

Figure 1. Example initial configuration and how it is represented in the text file.

*Draw the **class diagram** and develop a **java standalone application** to simulate this game. Don't forget to briefly document it. Do not create a GUI. All interaction is via* `System.in—System.out`*.*