Date, Group: Name, Neptun:

**Programming III.**

**ZHP-X**

We want to simulate a console flower garden, in which the flowers are represented by a single character, and every flower goes through the following stages: - 🡺 + 🡺 w 🡺 W 🡺 o 🡺O 🡺 X. The first stage (-) is the „seed”, the last one (X) is the fully opened flower, it will not grow any longer, it has to be collected. Implement the following classes:

1. Class Library: FlowerUtils – Console operations are NOT ALLOWED HERE!
	1. Flower class: stores the position and the current character of the flower.
	2. Garden class: stores a list made up from flowers and the position of the player. The player has to plant and collect flowers. This class should have the following methods:
	3. MovePlayer(int dx, int dy) method, that has to move the player with the specified vector (e.g. dx=0, dy=-1 🡺 move up). We should not be allowed to step off the playing field (the playing field is the console window – the size of this should be given to the Garden class using constructor parameters!)
	4. PlantFlower() method: if there is no flower in the current location of the player, a new seed is placed in the location of the player. For the new seed, start a new Task, that has to “grow” the flower, so the character representing the flower should be changing, one change in every seconds. The tasks should finish when the flower is fully opened (and also tasks should be capable of cancellation).
	5. CancelAll() method: it should ask the tasks to stop, and it should wait for the tasks to complete.
	6. CollectFlower() method: if there is a fully opened flower in the position of the player, then we collect the flower (remove it from the list), otherwise it should throw an exception.
	7. SaveFlowers() and LoadFlowers() method: using the XDocument class, these methods should save into and load from an XML file; the file should contain the status of the garden (the location and the stage of all flowers, and the location of the player). Watch out that stage change and save/load operations cannot be executed simultaneously at the same time!
2. Console app: the player should be controlled by a random number generator. While there is no keypress, the console screen should refresh once in every 100ms, and the player (represented by the text cursor in the console window) should be moved randomly, and a new flower should be placed once in every 5 seconds. After the loop, ask the tasks to cancel. Save the garden into the xml, and when the program restarts, load the state from the xml file.
3. Class Library: GardenTests. Tests for the Garden class
	1. Create a test with two test cases that the MovePlayer() method changes the player’s location in a good way.
	2. Create a test that after calling PlantFlower(), a new seed is placed into list with the player’s current location.
	3. Create a test that after calling PlantFlower(), nothing happens if the player’s position already has a flower.
	4. Create a test that calling CollectFlower() will collect (= remove) the flower in the player’s location.
	5. Create a test that calling CollectFlower() will throw an exception if there is no flower in the player’s location.