5. Practical Work: State Machine & Detailed Class Diagram; Python – Fish & Bear World

5.1. Modelling:

5.1.1. Create a **detailed** **Class diagram** (i.e. including attributes, methods, multiplicities, etc.) corresponding the group's chosen topic & environment and based on the simple class diagram developed in practical work #4. Also **do show four class associations** types (i.e. Regular, Generalization, Shared aggregation, Composition) in your detailed class diagram – BUT do show them only if they are naturally fitting in your class diagram (i.e. according to use cases and system description).

NOTE: if any of class association types is not naturally fitting in your detailed class diagram, THEN create one more class diagram (i.e. topic – up to your choice) with at least 4 classes and include all the special class association types which are not part of you detailed class diagram mentioned above.

5.1.2. Develop an accordant **State Machine Diagram** for one non-trivial object/class (i.e. found in your detailed class diagram in point 5.1.1. above) corresponding the group's chosen topic & environment and related also with in previous practical work #3 described use case(s).

5.1.3. Technical environment: use software tool chosen by the group itself.

5.1.4. Maximum points awarded: **7**

TASK: Using Visual Studio Code environment write Python program with latter mentioned requirements.

5.2.1. Use Fish and Bear simulated world and the accordant code (Fish, Bear, Plant and World classes) introduced in our lecture material (i.e. 5.2. Part material). **[a]** Modify the mainSimulation function to create two lists. One list will keep track of the number of fish that are alive in each time unit, and the other will keep track of the number of bears. When the simulation is done, this data should be written to a file. The file should have three columns: one column for the time, one for the number of fish, and one for the number of bears. **[b]** Two events can cause a Bear to die: starvation and energy level dropping to 0. Develop accordant energy representing instance variable in class Bear and set the initial energy level (i.e. integer). Update accordingly class Bear to decrease energy level if bear is breeding or moving, and increase energy level if bear is eating. Bear is dead if energy level drops below 0.

Load all the necessary classes and run the Fish and Bear simulation. In description document include runtime screenshots of above solutions.

5.2.2. Maximum points awarded: **8**

5.3. Format of solution to be submitted – use MS Word template (available in ViA Moodle) „_pymod2021_grupa00_pd0_dokuments_sablons.docx” and accordingly rename, e.g.: „_pymod2021_grupa0**1**_pd**5**_dokuments.docx” and save in .docx or .pdf format.

The file should containe **descriptions, screen-shots**; also **accordingly change title page**, **footer**, list of **content**, accordingly change part „**Document history**” and part „**Contacts and responsible person(s)**”. If there are no appendices, then delete this chapter. Add **Python code** .py file(s) in attachment of e-mail.

5.4. **Subject** field should contain: **PYMOD2021: pd5**

5.5. The practical work should be submitted in e-mail by 17.01.2022 23:59.