ECOSYSTEM

ANIMAL

I implemented an Abstract class called *ANIMAL.* Classes Fish , Otter , and Bear all inherits their common features such as **gender, age, strength, maxAge, birthRate, movement** (forward or backward), and **moved** (to check if they have moved in a simulation step) from *ANIMAL* class. Strength, and Movement are randomly generated..Each animal type is instantiated with its desired age., however there is a check, to ensure the age does not exceed the maximum age the animal can attain. Each class, FISH, OTTER and BEAR implements all abstract methods listed in *ANIMAL,* and inherits all already concrete methods in *ANIMAL.* A **toString()** method prints the label of each animal,

<<ECOSYSTEM>>

EcoSystem is an interface that advertises a number of methods as shown in the model below.

RIVER

River class, implements the interface Ecosystem. It is instantiated with the number of animals desired. An array of animals is then constructed. An animal can be added to the river by specifying the specie, and number of animals. Before an ***animal is added*** the following checks are made:

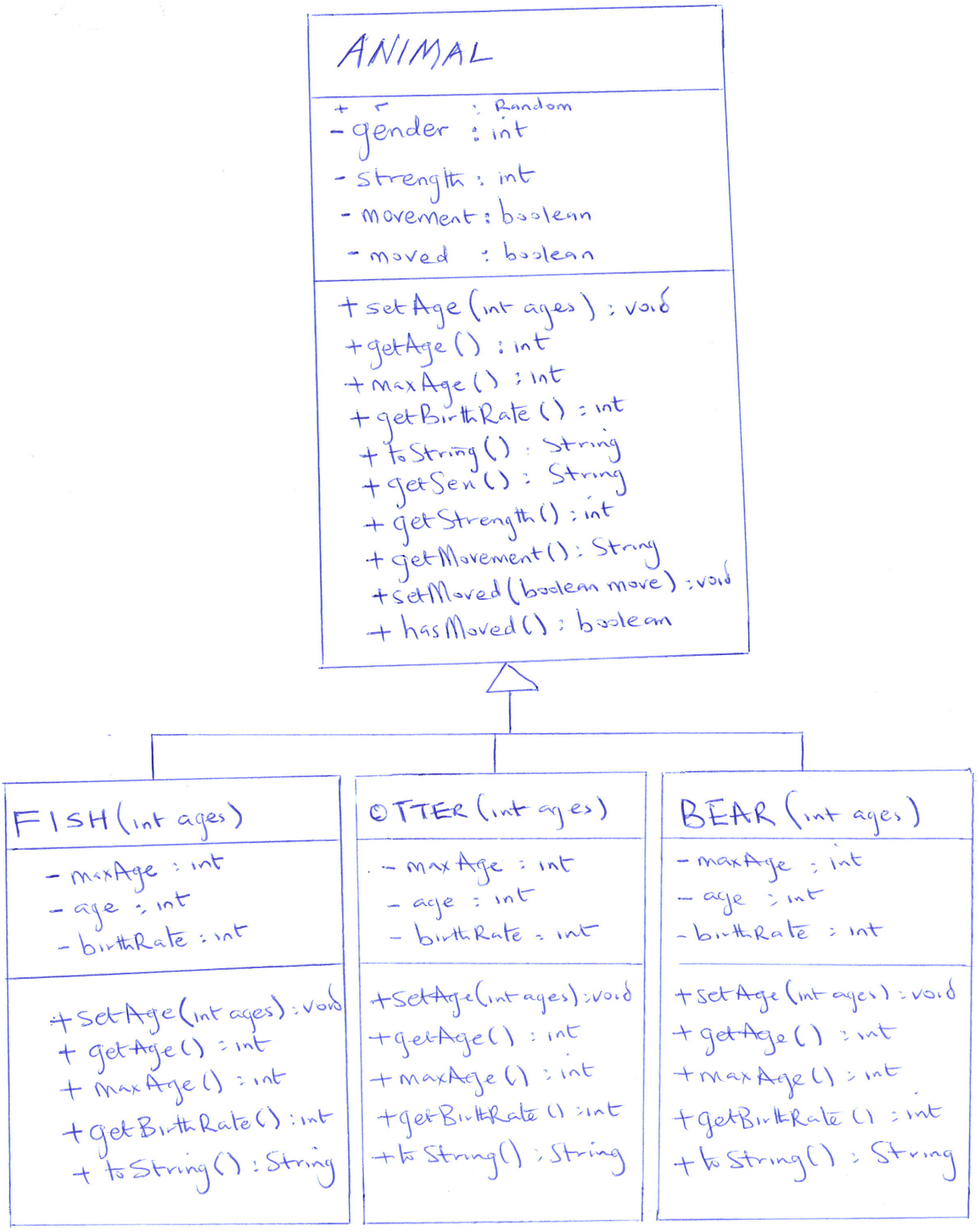
- Does the river have enough space to take in more animals? It returns 0, if NO, and returns the number of animals, say *x,* it can take in, if YES.
- Since each specie has a maximum space or population, a method checks if the river can take *x* number of the desired specie. It returns 0, or the number of animals of that specie that can be absorbed.
- A random number between 0 and the river size is generated..the cell is then checked if free..if YES the animal is added in that cell, else the process of random number generation continues.

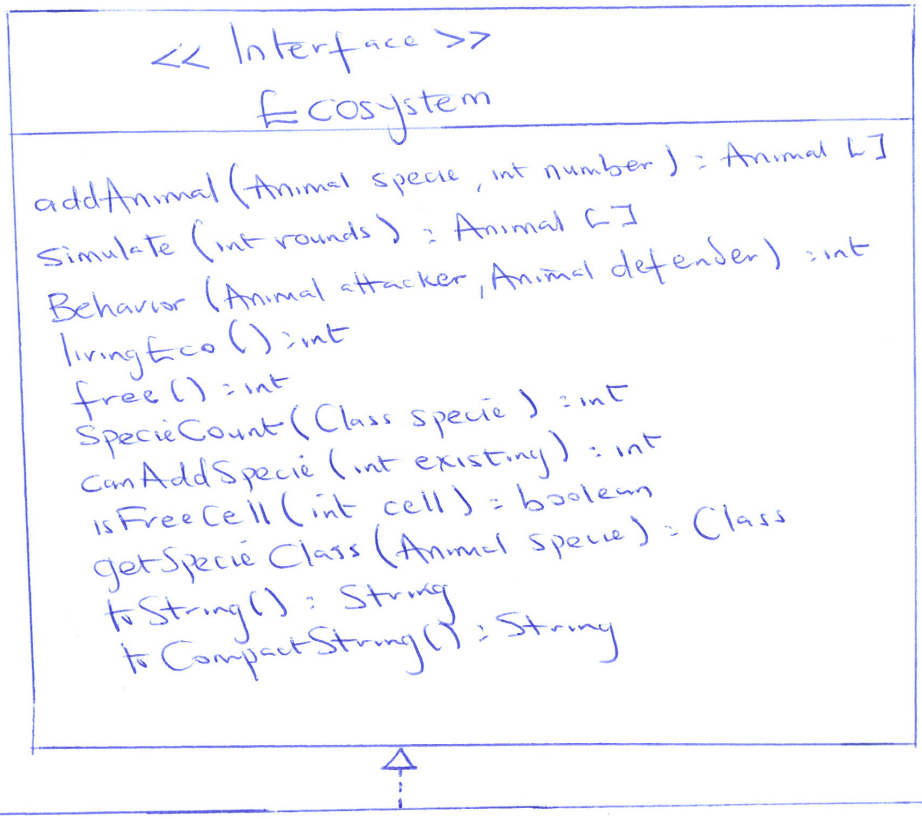
Simulation

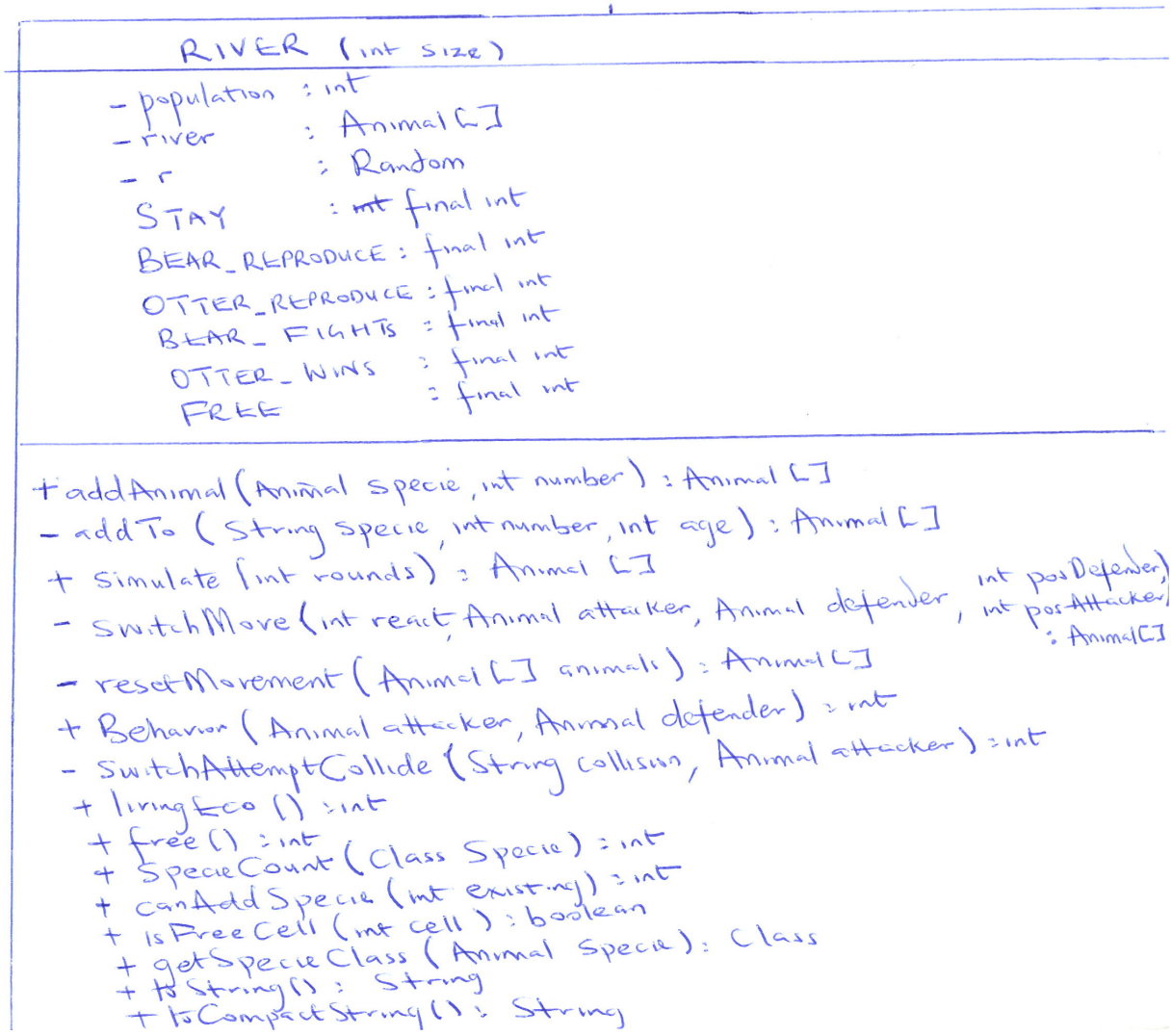
A simulation of the system can be run for *n* rounds. In each round, a check is done to ensure that an animal has not already moved once..after a round of simulation every animal is set to freely move again. The following steps are implemented in the simulation process:

- Check that the animal is alive and the animal has not moved.
- The animal being processed is the attacker, and its position is stored.
- The direction(movement) of the attacker is called..if "forward" movement is returned, then the next cell is the defender..or the previous cell in the case of a backward movement. However an extra check is called if the attacker is positioned at the end of the array and is trying to move forward(then the defender is the first cell) but if positioned in the first cell and is trying to move backward then the defender is the last cell of the array.
- These two animals, attacker and defender are sent to a method **Behavior(..),**that simply returns an integer, *s* representing their reaction as listed in the specification.
- Based on the reaction, *s,* a Switch Statement is used to implement the movement, or collision reaction.
- A check is performed if an animal has been killed already, if not, the age is increased (the **getAge()** is called and compared with the **maxAge().**If equal the animal dies, else the age is increased by 1) and the **hasMoved()** method is set to true.

A **toString()** method overrides the toString() method in each animal class and prints detailed information of each animal. A **toCompactString()** prints the general information of the state of the river.


ANIMAL
+ r : Random
- gender : int
- strength : int
- movement : boolean
- moved : boolean
+ setAge(int ages) : void
+ getAge() : int
+ maxAge() : int
+ getBirthRate() : int
+ toString() : String
+ getSex() : String
+ getStrength() : int
+ getMovement() : String
+ setMoved(boolean move) : void
+ hasMoved() : boolean
FISH(int ages)
- maxAge : int
- age : int
- birthRate : int
+ setAge(int ages) : void
+ getAge() : int
+ maxAge() : int
+ getBirthRate() : int
+ toString() : String
OTTER(int ages)
- maxAge : int
- age : int
- birthRate : int
+ setAge(int ages) : void
+ getAge() : int
+ maxAge() : int
+ getBirthRate() : int
+ toString() : String
BEAR(int ages)
- maxAge : int
- age : int
- birthRate : int
+ setAge(int ages) : void
+ getAge() : int
+ maxAge() : int
+ getBirthRate() : int
+ toString() : String


<< Interface >>
Ecosystem
addAnimal(Animal specie, int number) : Animal[]
Simulate(int rounds) : Animal[]
Behavior(Animal attacker, Animal defender) : int
livingEco() : int
free() : int
SpecieCount(Class specie) : int
canAddSpecie(int existing) : int
isFreeCell(int cell) : boolean
getSpecieClass(Animal specie) : Class
toString() : String
toCompactString() : String


RIVER (int size)
- population : int
- river : Animal[]
- r : Random
STAY : final int
BEAR_REPRODUCE : final int
OTTER_REPRODUCE : final int
BEAR_FIGHTS : final int
OTTER_WINS : final int
FREE : final int
+ addAnimal(Animal specie, int number) : Animal[]
- addTo(String specie, int number, int age) : Animal[]
+ Simulate(int rounds) : Animal[]
- switchMove(int react, Animal attacker, Animal defender, int posDefender, int posAttacker) : Animal[]
- resetMovement(Animal[] animals) : Animal[]
+ Behavior(Animal attacker, Animal defender) : int
- switchAttemptCollide(String collision, Animal attacker) : int
+ livingEco() : int
+ free() : int
+ SpecieCount(Class Specie) : int
+ canAddSpecie(int existing) : int
+ isFreeCell(int cell) : boolean
+ getSpecieClass(Animal Specie) : Class
+ toString() : String
+ toCompactString() : String