Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação com Interfaces Gráficas
Gabarito AD1 1^o semestre de 2018.
Tutor: Heder
Professores: Mario Benevides e Paulo Roma

---

1 AD1: River Simulation

O objetivo deste trabalho é escrever um programa em Python para simular um ecossistema, que consiste de ursos e peixes, vivendo em um rio. A simulação prossegue em ciclos, nos quais os animais envelhecem, um subconjunto deles migra para outros locais, outros se acasalam e alguns morrem, de acordo com certas regras.

2 Tarefas

Sua tarefa é implementar a classe Animal, bem como as subclasses Bear e Fish, e a class River. O método principal deve estar na classe RiverSimulator, que simula repetidamente as evoluções do ecossistema do rio. A cada iteração, RiverSimulator faz o seguinte:

1. Pede ao usuário como o ecossistema inicial do rio será gerado. Há duas opções:
   - Aleatório: neste caso, pede o comprimento do rio. Então, para cada célula, coloca um urso, um peixe, ou nulos de maneira uniformemente aleatória (i.e., com probabilidade igual). O gênero e idade de cada animal são escolhidos também de forma aleatória, garantindo que a idade de um animal esteja no intervalo apropriado (0 a 9 para um urso, 0 a 4 para um peixe).
   - Arquivo: Neste caso, pede o nome, e lê os dados de um arquivo. O formato da entrada está descrito abaixo:

2. Pede ao usuário o número de ciclos da simulação. Se o usuário fornecer zero ou um número negativo, o código não deve fazer nada, e esperar por um valor positivo.

3. Executa o número de ciclos requisitado, imprimindo o rio após cada passo.

Escreva e submeta testes para todas as classes requisitadas pelo projeto. Forneça código que funcione para todas as classes e métodos deste projeto.

3 Classes

A seguir, temos as classes que precisam ser entregues nesse projeto.

Animal	The abstract class Animal
Gender	The gender of an animal
Animal	An abstract, Animal, class that is used to reduce code repetition between similar characteristics of different animals
Bear	The concrete class Bear
Bear	The Bear class models bears
Fish	The concrete class Fish
Fish	The Fish class models fishes
River	The concrete class River
River	A river is represented by a string that looks like this:
RiverSimulator	Simulates an ecosystem that consists of bears and fish, living in a river
RiverSimulator	Implements a single method main for performing the dialogue with the user
RiverTest	Unit test for certain aspects of the behavior of RiverSimulator

Figura 1: Classes do projeto

4 Exemplo

Na correção será executado o arquivo RiverSimulator.py, com a execução do mesmo é esperado a seguinte saída.

```
River Ecosystem Simulator
keys: 1 (random river) 2 (file input) 3 (exit)
1
Trial 1: 1
```

```
Random River
Enter river length: 8
Enter the number of cycles: 6
Initial river (seed = 3144729332):
--- --- FM4 FF4 BF0 --- --- BF9

After cycle 1 :
--- --- --- --- --- --- --- BF1

After cycle 2 :
--- --- --- --- --- --- --- BF2

After cycle 3 :
BF3 --- --- --- --- --- --- ---

After cycle 4 :
BF4 --- --- --- --- --- --- ---

After cycle 5 :
BF5 --- --- --- --- --- --- ---

After cycle 6 :
--- --- --- --- --- --- BF6 ---

Final river:
--- --- --- --- --- BF7 --- ---

River Ecosystem Simulator
keys: 1 (random river) 2 (file input) 3 (exit)
```

5 Implementação dos métodos

Abaixo estão listados os métodos que você deve implementar para cada classe, alguns métodos são fornecidas como exemplo baseado em algumas dúvidas que foram identificadas durante tutoria, não será fornecido o código completo da atividade, o mesmo ainda será continuado no decorrer da AD2.

5.1 A classe Animal

A classe abstrata Animal é usada para reduzir a repetição do código entre características semelhantes de diferentes animais.

```
from abc import ABCMeta, abstractmethod

from enum import Enum
##
 # Enumerated type representing possible genders
 # of animals in the River simulation.
 #
class Gender(Enum):
    FEMALE = 1
    MALE = 2

class Animal(object)
    def __init__(self,age=None,gender=None):
        """Create an animal of the specified age and gender.
        If no age and gender are passed, then create
        an animal of a random age and gender."""
        if gender is None:
            self.gender = Gender.FEMALE if random.randint(0,1) == 0 \
                                        else Gender.MALE
        else:
            self.gender = gender
        self.age = age
        ....


@abstractmethod
```

```
def getAge(self):
        """Get the age of the animal."""
        pass

@abstractmethod
def maxAge(self):
        """Returns true if the current age of the animal
        has reached the limit for the species.
        Otherwise, it returns false."""
        pass

@abstractmethod
def incrAge(self):
        """If the current age of the animal is less than the maximum
        for the species, increments the age of the animal by one
        and returns true.
        Otherwise, it leaves the age as is and returns false."""
        pass
```

5.2 A classe Fish

Esta classe estende a classe Animal, provendo implementações apropriadas do construtor, getAge(), maxAge(), e incrAge().

```
from Animal import *
class Fish (Animal):

def incrAge(self):
        limit = False
        if not self.maxAge():
                self.age += 1
                limit = True
        return limit

def __repr__(self):
        fish = "F"
        fish += "F" if self.gender == Gender.FEMALE else "M"
        fish += str(self.getAge())
        return fish

def getStrength(self):
        """Get the current strength of the bear."""
```

5.3 A classe Bear

Esta classe estende a classe Animal, provendo implementações apropriadas do construtor, getAge(), maxAge(), e incrAge(). Adicionalmente, ela oferece mais um método novo (getStrength). A classe Bear modela um urso. Um urso vive até o final do seu 9º ano (10 ciclos de simulação), a menos que seja morto anteriormente. Um urso tem uma força que varia de acordo com a idade.

```
from Animal import *
class Bear (Animal):

def __repr__(self):
        bear = "B"
        bear += "F" if self.gender == Gender.FEMALE else "M"
```

```
        bear += str(self.getAge())
        return bear


def getStrength(self):
        """Get the current strength of the bear."""
```

5.4 A classe River

Repare que alguns métodos abaixo permitem que você crie uma semente para o gerador de números aleatórios. Isto pode ser útil nos seus testes, porque torna mais fácil replicar os resultados.

```
from Animal import *
from Bear import *
from Fish import *

class River(object):

def __init__(self, arg, seed=None):
        """If arg is a string, then it will be the inputFileName,
        and the river will be constructed from a file.
        If arg is an integer, then it will be the length of the river.
        If seed is not None, then generates a random river ecosystem
        of the given length, where the seed of the random number
        generator is as given.
        May raise IOError"""

        self.river = []  # lista de animais
        ## The length of the river (number of cells).
        self.length = 0


def getLength(self):
        """Returns the length of the river"""

def setSeed(self, seed):
        """Sets the seed of the random number generator used in
```

```
        updating the river."""

    def numEmpty(self):
        """Returns the number of empty (None) cells in the river."""

    def addRandom(self,animal):
        """If the river has no empty (None) cells, then do nothing
        and return False.
        Otherwise, add an animal of age 0 of randomly chosen
        gender and of the same type as animal to a cell chosen
        uniformly at random from among the currently empty cells
        and return True."""

    def updateCell(self, i)
        """Process the object at cell i, following the rules given in
        the Project Description. If it is None, do nothing.
        If it is an animal and it has reached the end of its
        lifespan, the animal dies.
        Otherwise, decides which direction, if any, the animal
        should move, and what other actions to take
        (including the creation of a child),
        following the rules given in the Project Description."""

    def updateRiver(self):
        """Perform one cycle of the simulation, going through
        the cells of the river, updating ages, moving animals,
        creating animals, and killing animals, as explained
        in the Project Description."""
        for i in range(self.getLength()):
                if self.river[i] is not None:
                        if not self.river[i].incrAge():
                                self.river[i] = None   # die
        for i in range(self.getLength()):
                self.updateCell(i)

    def write(self,outputFileName)
        """Write the river to an output file.
        May raise IOError."""
```

```
def __repr__(self):
        """Produce a string representation of the river."""
        riverStr = ""
        for i in range(self.getLength()):
                riverStr += (River.cEmpty * 3) \
                    if self.river[i] is None else str(self.river[i])
                        if i < self.getLength()-1:
                                riverStr += " "
        return riverStr
```