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

AD1: River Simulation

1 Diretivas Importantes

- Este trabalho deve ser feito apenas por você. Se precisar de ajuda, procure um dos tutores presenciais ou à distância. Certifique-se de que você entende o significado do termo **desonestidade acadêmica**.
- Comece o trabalho tão logo quanto possível e obtenha respostas às suas perguntas o quanto antes.
- Leia esta especificação completamente antes de começar.
- É possível que alguns aspectos desta especificação mudem ao longo do tempo, caso alguma incorreção seja detectada.

1.1 Introdução

O objetivo deste trabalho é escrever um programa em Python para simular um ecosistema, 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. A seguir, será explicado como o rio e seus habitantes devem ser modelados.

O ecosistema do rio deve ser implementado através de uma classe River, cujas primeiras linhas são algo como[1]:

[1]Mais detalhes sobre esta e outras classes serão fornecidos ao longo deste projeto.

Age	0	1	2	3	4	5	6	7	8	9
Strength	1	2	3	4	5	4	3	2	1	0

Tabela 1: Força × Idade.

```
class River(object):
    __init__ (self,arg,seed=None):
        self.river = []  # lista de animais

    ....
```

A variável de objeto *river* representa um rio como uma lista, cujas células são referências para objetos pertencentes a classe Animal, que representa formas de vida (ursos e peixes).[2]

Uma célula pode ser nula, significando que não contém nem um urso ou um peixe. Animal é uma classe abstrata, cujos os métodos devem ser implementados nas suas subclasses. Cada animal possui um campo gênero e um campo idade:

```
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):
        if gender is None:
           self.gender = Gender.FEMALE if random.randint(0,1) == 0 \
                                       else Gender.MALE
        else:
           self.gender = gender
        self.age = age
        ....
```

A classe Animal possui duas subclasses:

- A classe Fish modela peixes. Um peixe vive no máximo até o final do seu quarto ano (ciclos de cinco anos), a menos que seja morto antes.
- A classe Bear modela ursos. Um urso vive até o final de seus 9 anos (ciclos de dez anos), a menos que seja morto antes. Um urso possui uma força que varia de acordo com a sua idade, da seguinte maneira 1:

[2]Observe-se que *river* é declarado público, ao invés de privado. Embora programação orientada a objeto aconselhe ao contrário, escolhemos desse modo por conveniência na hora de testar o programa. Faremos dessa forma para vários, senão a maioria, dos métodos e variáveis de instância (objeto) que sejam necessários implementar. É claro, que no mundo real, dos empregos, seria esperado que fosse utilizado encapsulamento, de forma que detalhes, como o rio, deveriam ser mantidos privados.

Como vai ser explicado mais tarde, a configuração inicial do rio pode ser gerada aleatoriamente ou pode ser fornecida pelo usuário. Depois disso, a simulação prossegue em ciclos; o número de ciclos também faz parte da entrada de dados. A cada ciclo, os animais do rio envelhecem um ano, e aqueles que excederem o seu tempo de vida previsto, morrem (desaparecem). As células do rio são tratadas da esquerda para a direita, começando na célula 0. Se a célula contiver um urso ou um peixe, este animal escolherá aleatoriamente ficar onde está, ou tentar mudar para uma célula adjacente à sua direita ou à sua esquerda. As regras para movimentação são as seguintes:

1. Se a célula para a qual o animal se mover estiver vazia, o animal é simplesmente movido para a nova célula.

2. Se um urso ou um peixe colidirem, o peixe morre, a despeito do gênero do animal.

3. Se dois animais da mesma espécie e gênero colidirem na mesma célula, então:

   (a) Se forem peixes, eles ficam onde estiverem, e nada acontece.

   (b) Se forem ursos do mesmo gênero e de mesma força, eles ficam onde estiverem, e nada acontece.

   (c) Se animais do mesmo gênero e forças distintas colidirem, o mais fraco morre.

4. Se animais do mesmo tipo, mas de sexos opostos, estiverem para colidir na mesma célula, eles ficam onde estiverem, mas é criada uma nova instância deste tipo de animal, que será colocada numa célula aleatória vazia (previamente nula) da lista. Se a lista estiver cheia, nenhum animal é criado.

Notas:

- Embora vários animais possam se mover em cada ciclo, eles são processados um por vez, da esquerda para direita, de forma que movimentos simultâneos não ocorrem. Por exemplo, suponha-se que haja duas ursas adjacentes de idade 1 e 2, representadas pela string "BF1"and "BF2", conforme descrito abaixo:
  . . . BF1 BF2. . .
  A simulação deve considerar BF1 antes de BF2. Se BF1 se mover para direita, ela será morta por BF2 antes que BF2 seja considerada para mover-se. Da mesma forma, se BF1 ficar parada, mas BF2 se mover para a esquerda, então, BF2 matará BF1. Note-se que, se movimentos simultâneos fossem permitidos (mas não são), BF1 e BF2 poderiam pular uma sobre a outra, evitando a morte.

- Para modelar o nascimento de um animal, usa-se instanciação de objeto.

- Quando um objeto animal é instanciado como resultado da regra 4, sua idade é zero e seu gênero é aleatório.

- Um movimento de um objeto Animal é modelado atualizando-se referências, por exemplo, para mover um peixe da célula 8 para a célula 9, fazemos a célula 9 apontar para este objeto, e a célula 8 fica nula.

- A morte de um animal é modelada removendo-se a referência para ele, tanto tornando a referência nula, como fazendo-a apontar para outro objeto qualquer.
- O rio deve ser criado de forma circular; ou seja, sua última e primeira células devem ser consideradas adjacentes. Por exemplo, suponha-se que a lista *river* tenha tamanho 12, e que river[11] referencie um objeto Bear, e que river[0] referencie um objeto Fish. Então, o urso decide mover-se para a direita, matar o peixe, deixando o urso na célula 0, e deixando a célula 11 nula.

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 ecosistema do rio. A cada iteração, RiverSimulator faz o seguinte:

1. Pede ao usuário como o ecosistema 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).[3]
   - 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.

Os métodos associados às várias classes estão descritos na seção 4.
Tarefas da Parte 1: Escreva e submeta testes para todas as classes requisitadas pelo projeto.
Tarefas da Parte 2: Forneça código que funcione para todas as classes e métodos deste projeto.

3 Exemplo

O rio é representado por uma string como esta:

```
--- FF2 BM3 --- BF1 BM8 FM0 --- --- BF4
```

No exemplo acima, o objeto River possui uma lista de comprimento 10. Cada elemento é representado por uma string de três caracteres, que são `'---'`, que significa nulo, ou tem a forma $\alpha\beta\gamma$, onde:

[3] Gere inteiros uniformemente distribuídos usando `randint()` ou `choice()`, do módulo random.

- α pode ser F ou B, dependendo se a célula referencia um Fish ou um Bear,
- β pode ser F ou M, dependendo se o objeto referenciado é macho ou fêmea, e
- γ é um único dígito, entre 0 e 4 para um peixe e entre 0 e 9 para um urso, que fornece a idade do objeto.

Há exatamente um espaço separando cada string de três letras.

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

Trial 1: 1
Random river
Enter river length: 8
Enter the number of cycles: 1
Initial river:

BF3 FF4 --- --- BM2 BF9 --- FM3

After cycle 1:

FM4 BF4 --- BF0 BM3 --- --- ---

Trial 2: 1
Random river
Enter river length: 11
Enter the number of cycles: 3
Initial river:

--- FF1 FM3 --- --- --- BF3 --- --- BF2 ---

After cycle 1:

--- FF2 FM4 --- FM0 --- --- BF4 --- BF3 ---

After cycle 2:

FF3 --- --- --- FM1 --- --- --- BF5 BF4 ---

After cycle 3:

--- --- --- --- FN2 --- --- --- --- BF6 FF4

Final river:

FM4 BF4 --- BF0 BM3 --- --- ---

...

Trial 4: 3
```

Nota:

- O seu código deve imprimir as mesmas mensagens de texto para mostrar as interações do usuário.
- Deve-se implementar o método `__repr__(self)` da classe River para produzir exatamente esta representação do objeto River.

4 Métodos

Abaixo estão listados os métodos que você deve implementar para cada classe. Como boa parte dos testes é automatizada, você deve seguir rigorosamente as especificações dadas aqui. Isto significa, entre outras coisas, que nomes de classes, variáveis de instância e métodos devem ser mantidos. É claro que você pode definir seus próprios métodos auxiliares.

4.1 A classe Animal

```
try:                  # python >= 3.4
   from abc import ABC, ABCMeta, abstractmethod
except ImportError:  # python 2
   from abc import ABCMeta, abstractmethod
   ABC = object

class Animal(ABC):
    __metaclass__ = ABCMeta
    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."""

    @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
```

Animais devem possuir um método `__repr__(self)`, que retorna, por exemplo, 'BF7' para uma ursa de 7 anos.

4.2 A classe Fish

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

4.3 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:

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

4.4 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.

```
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"""

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."""

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

def __repr__(self):
        """Produce a string representation of the river."""
```

5 Estilo e Documentação

Mais ou menos 15 a 20% dos pontos será pela documentação e estilo de programação. Parte dos pontos estará baseado em quão bem for empregada herança, para reduzir a quantidade de código duplicado.

- Cada classe, método, e variáveis de instância (objeto), tanto públicas como privadas, devem ter um comentário que faça sentido. A documentação do projeto deve incluir o tag @author, e métodos devem incluir os tags @param e @return, apropriadamente.
- Todos os nomes de variáveis devem ter algum significado (i.e., de acordo com o valor que armazenam).
- Use variáveis de instância somente para os estados permanentes do objeto. Use variáveis locais para cálculos temporários dentro dos métodos. Todas as variáveis de instância devem ser privadas.
- Use um estilo consistente para identação e formatação.

6 O que entregar

Por favor, submeta um arquivo .tar.gz ou zip que inclua todos os arquivos utilizados. No arquivo comprimido, todos os arquivos devem estar dentro de um diretório chamado AD1. Se houver arquivos extra, como classes de teste, estes podem ser incluídos também. Normalmente, a coisa mais simples a fazer é clicar com o botão direito do mouse no diretório fonte do seu projeto e selecionar "Send To → Compressed/zipped file". Após criar o aquivo zip,

verifique-o cuidadosamente. Extraia os arquivos em um diretório temporário vazio, e olhe-os antes de submetê-los. Eles são arquivos .py? Todos os arquivos necessários estão presentes no arquivo comprimido? Estão nos diretórios corretos? Fazem parte da última versão que funciona do seu código?