**Evolution Emulator**

This project tries to emulate natural evolution in a simplistic manner. It is made to see the patterns that arise while different individuals from different species reproduce in an environment giving rise to new genetic patterns. The user needs to input a text file with some data that is mandatory for executing the project.

Still in its initial phase, some of the classes and logic that is thought of till now is subject to change if it is necessary.

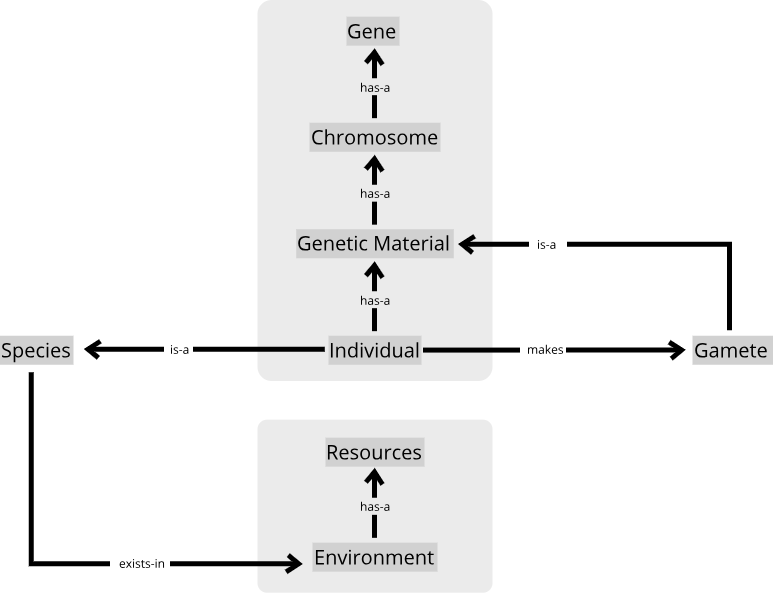
**Background**

1. **Earliest Life form**It is speculated that all living organisms in the world are direct descendants of only one species that existed on Earth during the dawn of life. This organism would most probably be a single celled organism that is speculated to have inhabited around deep sea hydrothermal vents, extracting their nutrients from the minerals that were coming out of the Earth’s crust. This organism is given the name LUCA or Last Universal Common Ancestor.
2. **Genetic information**Genetic Information of all living beings are stored in their DNA. DNA is like a code that determines the characteristics of the organism.  
   However, DNA by itself does not code for anything. It is at the base of all the layers that decide the characteristic of the organism. A DNA unit is made of pentose carbohydrate molecules that have a phosphate linkage and a base attached to it. Multiple DNA units combine to form a polymer that we call the DNA strand.  
   There are five nucleobases - adenine (A), cytosine (C), guanine (G), thymine (T), and uracil (U). Uracil however is not present in DNA but is present in RNA which also codes for characteristics of the organism. Thus, a DNA strand can be represented as the initials of the bases that are present in it. For example, ATGTCATGA. (Although real world DNA is never that small).  
   A group of three bases represents a codon. A codon codes for one amino acid.  
   Multiple codons together form a gene. A single gene codes for one protein which is a polymer of amino acids.  
   Some codons like ATG act as the indication that a new gene is being started, while others like TAG acts as the indication that a gene has ended.  
   Since all of our bodily functions are governed primarily by proteins, they determine the further characteristics of the organism.  
   For example, Hemoglobin, the protein that enables Oxygen transport through the blood, determines certain characteristics of the living beings that contain it. Like the color of our blood which is a characteristic of the being is determined by the protein Hemoglobin. Also it determines which gases are toxic to us, as those gases that may form a stable compound with Hemoglobin than Oxygen (like Carbon Monoxide) will cut off Oxygen supply, thus suffocating the organism to death.
3. **Reproduction**There are two modes of reproduction - Sexual and Assexual. Inside the nucleus of any cell there are chromosomes. These chromosomes contain the genes. Chromosomes exist in pairs. Since they carry genes, each chromosome is deterministic of certain characteristics of the individual organism they are a part of.  
   **Assexual Reproduction**During this mode of reproduction, the chromosome pairs duplicate, thus duplicating the DNA. The cell itself divides into two new cells and each new cell gets one copy of the original chromosome. Variation in characteristics in this mode of reproduction arises from random mutations or errors in duplicating.  
   **Sexual Reproduction**During this mode of reproduction the chromosome pairs disintegrate into singular chromosomes. The new cells formed here are called gametes. Each gamete only comprises half of the genetic information. Two gametes from two different individuals combine together to form a complete cell (individual). Variation in characteristics in this mode of reproduction arises from the intermixing of the genome along with random mutations.
4. **Biosphere**From a biological perspective, every environmental region is divided into different biomes. Each biome contains many ecological niches that are like opportunities for life to thrive. When in a biome, if certain ecological niches are deprived of life, then organisms from other ecological niches have a high chance to evolve and fill these empty niches. This is also known as Divergent Evolution.  
   Every opportunity for life is due to the presence of certain resources available in abundance. However all resources chain back to sunlight and the various other minerals that are important for life to sustain.

**Project Structure**Object Oriented approach is used to model the various components of this project.

Since modelling every protein and gene is cumbersome and out of scope, a simple mapping of characteristics to the gene is not only sufficient but also easier.

A diagram denoting the basic structure of the project and the relation between various classes is given below :



Genes are modelled using Strings. Each gene is extracted from the chromosomes. The extraction of genes is as per the fact that a specific codon (the start codon) denotes the starting of a gene while specific codon(s) (the stop codon(s)) denote the ending of the gene.

Example :

Let’s say there are two chromosomes (each chromosome has two pairs of strands) :

*GA****ATGTGCTCCCTAATGTGA****C* , ***ATGTGCAATAGTATGTGA***

And

G***ATGCGTCGACGTTGA****CGT* , *C****ATGCAACGTTAGTGA***

The parts in bold represent a gene. Multiple chromosomes can be present in the genetic material. The encoding of these genes, i.e., how they would decide the characteristics is yet not prepared.

The environment is to be modelled using a 2-D matrix, where each cell represents an ecological niche. Some fundamental resources would be initialized by the user, which would then be distributed into the environment (that is the cells of the matrix) in a random fashion. Each niche will have some reserves of each of the resources (some may be absent too). The absence of a resource in the niche would mean that no amount of uncertainty would ever result in creating that resource in that niche.

How and which organism would use a particular resource would be encoded in the genes, which is yet not decided.

**Documentation**

**environment.data.Environment**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | int | rows |
| int | cols |
| ArrayList<Resources>[][] | environment |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Environment(int rows, int cols)** |

**environment.data.Resources**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | class | Resource |
| ArrayList<Resource> | resources |
| float | replenishFactor |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Resources()** |
| **Resources(float replenishFactor, HashMap<Integer, Integer> resources)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **public** | void | **depleteAllResources(int depleteBy)**  Deplete the available amount of all the resources by the amount ‘depleteBy’. |
| void | **depleteResource(int ind, int depleteBy)**  Deplete the available amount of the resource at the given index ‘ind’ by the amount ‘depleteBy’. |
| void | **replenishAllResources()**  Replenish the available amount of all the resources. |
| void | **replenishResource(int ind)**  Replenish the available amount of the resource at the given index ‘ind’. |
| void | **setAllResources(int amount)**  Set the amount of all the resources to the given amount. |
| void | **setResource(int ind, int amount)**  Set the amount of the resource at the index ‘ind’ to the given amount. |
| void | **setResources(float replenishFactor, HashMap<Integer, Integer> resources)**  Sets the fields of the object according to the given parameters. |

**Member class : private Resource**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | *final* int | resourceIndex |
| int | availableAmount |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Resource(int availableAmount, int resourceIndex)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **public** | void | **setAvailableAmount(int availableAmount)**  Setter for available amount. |
| int | **getAvailableAmount()**  Getter for the available amount. |
| int | **getResourceIndex()**  Getter for resource index. |
| void | **depleteAvailableAmount(int depleteBy)**  Depletes the available amount by the amount ‘depleteBy. |

**genetic.data.UniversalConstantParameters**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | *static* int | codonLength |
| *static* char[] | bases |
| *static* double | thresholdForNewSpecies |
| *static* double | probabilityOfMutation |
| *static* String | startCodon |
| *static* String[] | stopCodon |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **UniversalConstantParameters(*-ALL THE FIELDS-*)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **private** | String[] | **parseStringWithCommas(String string)**  Parses the given string and extracts different strings separated by commas. |

**genetic.data.Gene**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | ArrayList<String> | genes |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Gene()** |
| **Gene(String chromosomeStrand)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **private** | String | **findGene(int cursor, String chromosomeStrand)**  Finds all the genes that are present in the chromosome strand string. This doesn't store these genes. |
| int | **findStartCodon(int startIndex, String chromosomeStrand)**  Finds and returns the index of the start codon in the chromosome strand string starting from the startIndex. |
| int | **findStopCodon(int startIndex, String stopCodon, String chromosomeStrand)**  Finds and returns the index of the provided string (StopCodon) in the chromosome strand string starting from the startIndex. |
| int[] | **getAllStopCodonIndicies(int startIndex, String chromosomeStrand)**  Returns an array containing the indices of stop codons that are found after the startIndex. |
| void | **storeGenes(String chromosomeStrand)**  Stores all the genes into the genes field. |
| **public** | void | **setGenes(String chromosomeStrand)**  Finds and stores the genes from the chromosome strand string. |

**genetic.data.Chromosome**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | String | chromosomeStrand1 |
| String | chromosomeStrand2 |
| Gene | genesOnStrand1 |
| Gene | genesOnStrand2 |
| ArrayList<Character> | bases |
| HashMap<Character,Double> | percentageOfBase |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Chromosome()** |
| **Chromosome(String strand1, String strand2)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **private** | String | **changeAtIndices(int[] indices, String string, char ch)**  Changes the i’th character of the given string to ‘ch’ for each index ‘i’ in the array indices. |
| ***static*** double | **countDifference(...)**  *TODO...* |
| double | **noOfOccurrence(char ch, String string)**  Counts the number of occurrences of ‘ch’ in the given string. |
| **public** | void | **addRandomBase(char base)**  Adds the given ‘base’ in a randomly selected chromosome strand field at a randomly selected index. Note that while selecting a strand, this method can randomly select either one of them or may select both of them, with each of the three cases having a probability of 33.33% of occurring. |
| *static* double | **calculateGeneticChange(...)**  *TODO* |
| void | **changeRandomBase(char base, int numberOfChangedBases)**  Changes a randomly selected base(character) from a randomly selected chromosome strand. Note that while selecting a strand, this method can randomly select either one of them or may select both of them, with each of the three cases having a probability of 33.33% of occurring. |
| String | **getOneRandomChromosomeStrand()**  Returns one of the two chromosome strands at random. |
| void | **removeRandomBase(char base)**  Removes a randomly selected base(character) from a randomly selected chromosome strand. Note that while selecting a strand, this method can randomly select either one of them or may select both of them, with each of the three cases having a probability of 33.33% of occurring. |
| void | **setBaseList()**  Stores the individual bases(characters) that are present in both of the chromosome strands. |
| void | **setChromosome(String strand1, String strand2)**  Stores the chromosome strands and finds the genes present on each of them. |
| void | **setPercentageOfBase()**  Stores the percentage of each base(character) present in the chromosome strands. |
| String | **toString()** |

**genetic.data.GeneticMaterial**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | ArrayList<String> | chromosomes |
| **protected** | int | noOfChromosomes |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **GeneticMaterial()** |
| **GeneticMaterial(Chromosome[] chromosomes)** |
| **GeneticMaterial(ArrayList<Chromosome> chromosomes)** |
| **GeneticMaterial(GeneticMaterial geneticMaterial)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **public** | void | **setGenes(String chromosomeStrand)**  Finds and stores the genes from the chromosome strand string. |
| *static* double | **calculateGeneticChange(...)**  *TODO* |
| void | **makeRandomMutation(int noOfChanges)**  Makes one of these changes to the one of the chromosomes selected at random : add a base, change a base, remove a base. |
| void | **setCharacteristic(chromosome[] chromosomes)**  Sets all the fields. |
| String | **toString()** |

**genetic.data.Gamete**

extends genetic.data.GeneticMaterial

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | String[] | gameteStrand |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Gamete(Chromosome[] chromosomes)**  Creates a Gamete object with the given chromosome array. |
| **Gamete(GeneticMaterial geneticMaterial)**  Creates a Gamete object with the given genetic material. |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **private** | void | **makeGametes()** |
| **public** | *static* GeneticMaterial | **makeNewGeneticMaterial(Gamete g1, Gamete g2)** |
| String | **toString()** |

**genetic.data.Individual**

|  |  |  |
| --- | --- | --- |
| **Fields** | | |
| **private** | GeneticMaterial | geneticMaterial |
| int | speciesCode |
| double | percentageOfChange |

|  |  |
| --- | --- |
| **Constructors** | |
| **public** | **Individual(GeneticMaterial geneticMaterial)** |

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **public** | *static* Individual | **reproduce(Individual i1, Individual i2)**  Makes the two Individuals reproduce and creates their offspring (new individual), and returns this offspring. |
| void | **calculateChange(...)**  *TODO* |
| String | **toString()** |

**Reader**

This class reads a text file that contains necessary details that the user must input. This data is written in a simple format, as given below :

|  |
| --- |
| "codon-length" : "2"  "bases" : "ATGCU"  "threshold-for-new-species" : "5"  "probability-of-mutation" : "0.2"  "start-codon" : "ATG"  "stop-codon" : "TAA, TGA, TAG" |

After reading this data, this class extracts the necessary data from it. The separator in the middle ‘:’ is not fixed, and can be any comfortable separator.

|  |  |  |
| --- | --- | --- |
| **Methods** | | |
| **private** | int | **findSeperatorIndex(String dataCodeLine)**  Identifies and returns the index of the separator for the given line. |
| int | **setAllParameters(HashMap<String,String> items)**  Sets all the parameters as specified in the text file. |
| **public** | int | **readData(String fileName)**  Reads the text from the file as specified by the parameter. |
| int | **validateAndStoreData(ArrayList<String> dataCode)**  Validates that the format of the file is correct and then extracts the necessary data. |