Class Outline

**GeneticsSim**

Programming Assignment 1

February 9, 2016

**Prepared by**

Marcus Posey

[mlp0027@uah.edu](mailto:mlp0027@uah.edu)

**Prepared for**

Dr. Rick Coleman

CS 307, Object Oriented Programming

Computer Science Department

University of Alabama in Huntsville

Table of Contents

1. System Overview ……………………….. 1
2. Relevant Terms and Acronyms …………. 1
3. Object Description ……………………… 1
   1. Class Simulation …………………… 1
   2. Class Allele ………………………… 2
   3. Class Gene …………………………. 2
   4. Class Organism …………………….. 3
   5. Class ColemanXMLParser ………… 4
   6. Class Observable …………………... 4
   7. Class IObserver ……………………. 5
   8. Class LoveChamber ……………….. 5
   9. Class StatCounter ………………….. 5
4. **System Overview**

The main objective of the project is to produce a program capable of simulating Mendelian laws of inheritance. These laws of inheritance govern what composes the genetic makeup of the offspring of two organisms. This task will be achieved by creating software models of basic components of a living organism. These include, but may not be limited to, alleles, genes, genotypes, and the organism itself. By crossing the genotypes of two parent organisms that are to be read from an initial xml file, new genotypes that belong to offspring will be created. This data will then be recorded to acquire statistical representations of total gene and genotype counts among all children. All genetic data (i.e. that of parent organisms and offspring) will be presented on the screen in some manner—initially through a command-line interface and, in the future, possibly through a GUI.

1. **Relevant Terms and Acronyms**

* **Allele** – These are represented in the model by a single character that is either uppercase or lowercase. Uppercase letters dominant traits, while lowercase represent recessive traits. Pairs of alleles make up genes.
* **Gene –** Genes are pairs of alleles that influence the phenotype (visible characteristics) of an organism.
* **Genotype –** The collection of genes an organism possesses that contribute to its traits and characteristics. Within this program, the genotype can be thought of as the essence of an organism.
* **unordered\_map –** This is the STL implementation of a dictionary using a hash table. It assigns certain values to unique keys.
* **unique\_ptr** – This is the STL implementation of a managed pointer. It implements RAII to acquire possession of memory on the heap and free said memory when the container goes out of scope.

1. **Object Description**
   1. Class *Simulation*
      1. Member Variables

* **\_loveChamber –** A unique\_ptr containing a LoveChamber object. This object handles mating between organisms within the simulation
* **\_statCounter –** a StatCounter object. It observes \_loveChamber and records/prints stats of produced offspring
* **\_offspringCount –** the number of offspring that will be created during the simulation
  + 1. Member Functions
       1. Function *Simulation*

**Actions Performed –** Constructs a Simulation object

**Arguments –** None

**Return Value –** None

* + - 1. Function *run*

**Actions Performed –** Calls \_loveChamber’s mate function and then prints the results when all offspring have been created

**Arguments –** None

**Return Value –** void

* + - 1. Function *init*

**Actions Performed –** Queries the user for the data file location and generation count and uses this data to instantiate \_loveChamber

**Arguments –** None

**Return Value –** void

* 1. Class *Allele*
     1. Member Variables
* **\_symbol –** a single character that represents part of a trait
* **\_description –** a description of what trait the allele represents
  + 1. Member Functions
       1. Function *Allele*

**Actions Performed –** constructs an allele object with an initial symbol and description

**Arguments:**

**char symbol –** the character representation of the allele

**string description –** a description of what the allele represents

**Return Value –** None

* + - 1. Function *isDominant*

**Actions Performed –** determines if the allele is dominant or not (recessive)

**Arguments –** None

**Return Value –** returns true if the allele is dominant

**There are also get functions for the member variables.**

* 1. Class *Gene*
     1. Member Variables
* **\_alleles –** a vector containing two Allele objects that make up the gene
* **­\_description –** a description of the gene
  + 1. Member Functions
       1. Function *Gene*

**Actions Performed –** constructs a Gene object with an initial allele pair and description

**Arguments –**

**Allele a1 –** an allele

**Allele a2 –** an allele

**string desc –** a description of the gene

**Return Value -** None

* + - 1. Function *getPhenotype*

**Actions Performed –** determines the visible allele trait

**Arguments –** None

**Return Value –** phenotype as a string

* + - 1. Function *getZygosity*

**Actions Performed –** determines the zygosity of the two alleles

**Arguments –** None

**Return Value –** the zygosity of the gene as a string

* + - 1. Function *getAllelesString*

**Actions Performed –** concatenates the two allele symbols

**Arguments –** None

**Return Value –** returns a string containing the concatenated allele symbols

* + - 1. Function *toString*

**Actions Performed –** concatenates the zygosity, phenotype, and alleles into one string

**Arguments –** None

**Return Value –** returns a string containing the zygosity, phenotype, and alleles of the gene

**There are also get functions for the member variables.**

* 1. Class *Organism*
     1. Member Variables
* **\_genus –** the genus of the organism
* **\_species –** the species of the organism
* **\_name –** the common name of the organism
* **\_genotype –** a vector of Gene objects that make up the organism
* **\_serveCounter –** indicates the index of the next gene to retrieve through ::serveGene
  + 1. Member Functions
       1. Function *Organism*

**Actions Performed –** constructs an organism object with an initial genus, species, and name but no genotype

**Arguments –**

**genus –** the genus of the organism

**species –** the species of the organism

**name –** the common name of the organism

**Return Value –** None

* + - 1. Function *addGene*

**Actions Performed –** adds a Gene object to the \_genotype vector

**Arguments –**

**Gene gene –** the gene to add to the organism’s genotype

**Return Value –** void

* + - 1. Function *serveGene*

**Actions Performed –** retrieves the next Gene in \_genotype and increments \_serveCounter

**Arguments –** None

**Return Value –** returns the next Gene in \_genotype

* + - 1. Function *getGeneCount*

**Actions Performed/Return Value –** returns the number of genes in the genotype

**Arguments –** None

**There are also get functions for the member variables.**

* 1. Class *ColemanXMLParser*
     1. Member Variables
* **\_parser –** a GeneticsSimDataParser object. GeneticsSimDataParser is a class for parsing a nonstandard xml format. It was provided by the instructor
  + 1. Member Functions
       1. Function *ColemanXMLParser*

**Actions Performed –** instantiates the \_parser object

**Arguments –**

**string filename –** the name of the data file to parse

**Return Value –** None

* + - 1. Function *parseFile*

**Actions Performed –** turns the data in the file into two organisms and places them in a vector

**Arguments –**

**vector<Organism> organisms –** a vector for the two organisms the data file contains

**Return Value –** void

* 1. Class *Observable*
     1. Member Variables
* **\_observers –** a vector of IObservable objects
  + 1. Member Functions
       1. Function *addObserver*

**Actions Performed –**adds an observer to \_observers

**Arguments –**

**IObserver o –** an observer to add to the stored vector

**Return Value –** void

* + - 1. Function *removeObserver*

**Actions Performed –** removes an observer from \_observers

**Arguments –**

**IObserver o –** the observer to remove from the stored vector

**Return Value –** void

* + - 1. Function *notifyAll*

**Actions Performed –** sends the specified message to all observing objects

**Arguments –**

**T arg –** a message to send to all observers

**Return Value –** void

* 1. Class *IObserver*
     1. Member Variables

**None**

* + 1. Member Functions
       1. Function pure virtual *notify*

**Actions Performed –** processes some message

**Arguments –**

**T arg –** some message to process

**Return Value –** void

* 1. Class *LoveChamber*
     1. Member Variables
* **\_o1 –** a parent organism
* **\_o2 –** a parent organism
  + 1. Member Functions
       1. Funcion *LoveChamber*

**Actions Performed –** initializes the LoveChamber object starts a random time seed

**Arguments –**

**Organism o1 –** one parent organism

**Organism o2 –** one parent organism

**Return Value –** None

* + - 1. Function *mate*

**Actions Performed –** creates a specified number of offspring

**Arguments –**

**int count –** the number of offspring to generate

**Return Value –** void

* 1. Class *StatCounter*
     1. Member Variables
* **\_genotypeCounts –** an unordered\_map<string, int> that maps a genotype to the number of offspring who have that genotype
* **\_geneCounts –** an unordered\_map<string, unordered\_map<string, int>> that maps a gene description to a map of a gene string representation and its associated count
  + 1. Member Functions
       1. Function *notify*

**Actions Performed –** adds the occurring genotype and gene of the received organism to the stored counts

**Arguments –**

**Organism arg –** an organism with a genotype

**Return Value –** void

* + - 1. Function *printStats*

**Actions Performed –** prints the genotype and gene counts to the command window

**Arguments –** None

**Return Value -** void