FRAMEWORK

- **EvolutionaryEngine**
  - **EvolutionaryEngine**
    - An interface for the evolutionary process. Implementations will define the steps for evolution.
  - **Engine**
    - Concrete implementation of **EvolutionaryEngine**. This class is a Mediator that encapsulates the interaction between genotype constructors and expressions which create phenotypes. The phenotypes are manipulated by the **Engine** through Strategy to define how mating, mutation, and fitness selection are defined. It is also a director for the Builder pattern used by the **FirstGeneration** class.
- **Gene**
  - **Gene**
    - An interface for the properties a **Genotype** will be composed of. Implementations will define different traits for the **Phenotype** as properties.
  - **GeneFactory**
    - An interface for an implementation of a Abstract Factory which creates **Genes**. This allows any application to use specific **Gene** implementations in the framework.
  - **GeneticProperty**
    - An interface for metadata about a **Gene**. Can be used by the **Genotype** and the **GeneFactory** to interpret **Genes**.
  - **PropertyReport**
    - An interface to support storing **FitnessTest** results for a **GeneticProperty**
- **Genotype**
  - **Genotype**
    - An interface for a collection of **Genes**. Implementations will define different ways of storing the **Genes**.
  - **GenotypeList**
    - An linear list-based implementation of the **Genotype** interface..
  - **GenotypeTree**
    - A tree-based implementation of the **Genotype** interface.
  - **GeneIterator**
    - An interface for an implementation of the Iterator pattern for the **Genotype** interface.
- **GenerationBuilder**
  - **GenerationBuilder**

        - An interface for objects that create **Genotypes** using **Genes**. This uses the Builder pattern. This allows the framework to change how **Genotypes** are constructed.
    - **FirstGeneration**
        - An implementation of **GenerationBuilder** which represents creating **Genotypes** for the first generation of the evolutionary process. It is also an implementation of the Builder pattern which uses the **EvolutionaryEngine** as the director.
    - **GenerationCritera**
        - A container for parameters that the **FirstGeneration** will use.
- **Expression**
    - **Phenotype**
        - An interface for a representation of **Genotypes** as objects with behavior.
    - **ExpressionFactoryInterface**
        - An interface for an Abstract Factory that creates **Phenotypes** based on **Genotypes**. This allows any application to use specific **Phenotype** implementations in the framework.
    - **ExpressionFactory**
        - An implementation of **ExpressionFactoryInterface** as an Abstract Factory that binds **Genotypes** to **Priorities** and creates **Phenotypes** using them.
    - **PriorityInterface**
        - An interface for objects that attach weighted properties to **GeneticProperties**.
    - **Priority**
        - An implementation of **PriorityInterface** that represents a **GeneticProperty's** weight as an integer.
    - **PriorityFactoryInterface**
        - An interface for an Abstract Factory that creates **Priorities**. This allows any application to use specific **Priority** implementations in the framework.
    - **PriorityFactory**
        - An implementation of **PriorityFactoryInterface** as an Abstract Factory that uses the **Priority** implementation of **PriorityInterface**.

AI Game Solver

Objects to consider: RulesOfTheGame, Objectives, Genotypes, Expressions, Tasks, Player(Phenotype), Priority

I. **Rules and Tasks of the Game**

- There is a **RulesOfTheGameFactory** Factory
  - Uses the game Engine to incorporate Specifics of the Games Rules
  - **TaskRuleFactory**
    - Creates Tasks oriented Rules
    - Uses **GameEngine** to provide task related Objects
  - **StateRuleFactory**
    - Creates State oriented Rules
    - uses **GameEngine** to provide the game state
  - **TaskStateRuleFactory**
    - Creates Task Stated oriented Rules
    - uses **GameEngine** to provide game state and game interactive objects
- **Rules** are Products
  - These rules are created by their perspective Factory
  - Contains the specific task to complete the rules
  - **TaskRule**
  - **StateRule**
  - **TaskStateRule**
- **TaskBuilder** is a Builder
  - used to create Tasks from the **UserInputs**
  - uses a Template Method in its create() operation
  - **SingleBuilder**
    - Creates a SingleTask
  - **TransactionTaskBuilder**
    - Creates a TransactionTask
  - **MacroTaskBuilder**
    - Creates a MacroTask
  - **ChainTaskBuilder**
    - Creates a ChainTask
- **Tasks**
  - **SingleTask** is a Command
  - **ChainTask** is a Chain Of Responsibilities
  - **TransactionTasks** is a Command to back track the commands
    - only completed if it able to complete all Tasks within the Transaction, else it backtracks

    - **MacroTask** uses the Command Pattern
        - Uses multiple **SingleTask**'s to create a **MacroTask**

II. **Objectives**

- **ObjectiveBuilder** are Builders
    - used to create the Specific Objectives that are needed by the Phenotypes
    - create() method is a Template Method
    - uses addRule to build the Objectives, from the standPoint of the Rules
    - If the Rules add conflict, then create() will return False, else it will succeed
    - **StateObjectiveBuilder** creates StateObjectives
    - **TaskObjectiveBuilder** creates TaskObjectives
    - **TaskStateObjectiveBuilder** creates TaskStateObjectives
    - **CompositeObjectiveBuilder** creates CompositeObjectives
- **Objectives** is adapted from GeneticProperty from the Framework, It uses the Composite Pattern
    - **TaskObjective** is a leaf
    - **StateObjective** is a leaf
    - **TaskStateObjective** is a leaf
    - **CompositeObjective** Abstract class of the Objectives
        - **ListOfObjectives** this is the primary ObjectiveList
- **ObjectiveIterator** is an Iterator
    - **PreOrderIter** used by the ListOfObjectives to make a Iterator
- **ObjectiveVisitors** are Visitors
    - **ExecuteVisitor** is a Visitor
- **A Genotype** is a String representation of the RulesOfTheGame Just a Container of Genes
    - **GenotypePlayer**
        - represents a Phenotype as a Genotype. Interacts with **Expressions** through the **Player** and the **ApplicationEngine**
    - **GenotypeBuilder**
        - Builder pattern: This builds a genotype using **FirstGameGeneration** as the director. Allows for different construction of **Genotypes** by varying the **FirstGameGeneration**
- **Genes**
    - **GeneObjective**
        - This is the Gene of the Genotype that is represented from the Objective
    - **GeneFactory**
        - Abstract Factory pattern: This creates and distributes genes for the **GenotypeBuilder** to use. Separates creation of **Genes** from creation of **Genotypes**
- **Player**
    - **Player**
        - Composed by **Genotypes (GenotypePlayer)**, **Objectives**, and a **PriorityStrategy**. Uses the Strategy pattern to define gameplay

strategies using a **PriorityStrategy** to choose an objective. Is acted on by fitness, mating, and mutation through the **ApplicationEngine**.

    - **PriorityStrategy**
        - Strategy for a player. Encapsulates an algorithm for choosing an objective based on the **Objective**'s priority property. Attached to a player through the **PlayerFactory**. Can be extended to support different selection algorithms.

- **Expressions**
    - **PlayerExpressionFactory**
        - Abstract Factory pattern. Creates **Players** based on configurable **Objectives** and **Genotypes**. References **ObjectiveBuilder** and **PriorityFactory** (from framework).
- **Fitness**
    - **GameFitnessTest**
        - An implementation of FitnessTest from the framework. It is a Strategy for the **ApplicationEngine**. Defines an algorithm for determining the fitness score of a list of players, and will return a list of players that passed and should be mated based on implementation.
    - **Level**
        - An abstraction for a game level. Encapsulates the behavior for choosing the minimum fitness score a player needs to pass.
- **Mating**
    - **MatingPlayerStrategy**
        - A Strategy for the **ApplicationEngine** that encapsulates the behavior for creating a new **Genotypes** from two **Players**. The **ApplicationEngine** takes the **Genotype** and passes it to the **Expressions** system for the next generation.
- **Mutation**
    - **MutatePlayerStrategy**
        - A Strategy for the **ApplicationEngine** that encapsulates the behavior for creating a new **Genotype** which is a mutation of a **Player's Genotype**. The **ApplicationEngine** takes the **Genotype** and passes it to the **Expressions** system for the next generation.
- **GameGeneration**
    - creates the first generation for the game
- **GenerationCriteria**
    - used by the GameGeneration to create batch of Phenotypes
- **ApplicationEngine**
    - Mediator between rules, player, and game generation, and uses Strategy to define those interactions using the mating, mutation, fitness, and expression modules. Allows variations in how the different classes interact with each other without modifying class internals or knowing specifics about interactions.

UI Application Analyzer

- **GenotypeBuilder**
  - The GenotypeBuilder is Builder that constructs the genotype representation of the HTML and CSS for a Phenotype
  - **HTMLParser** is used by the GenotypeBuilder to parse the html and add it to the Genotypes genetic property Composite
  - **CSSParser** is also used by the GenotypeBuilder, and parses through the CSS, adding the appropriate styles to the DNA of the genes in the Composite

- **WebpageBuilder**
  - Uses a Builder to convert the composite Genotype back into it's phenotype representation.
- **Webpage**
  - This is a Product of the WebpageBuilder, it is the phenotype representation of the **WebpageGenotype**
- **WebpageMatchMaker**
  - This is a Strategy used in the Mating of two
- **GeneChanger**
  - The gene changer is a concrete Strategy which defines the algorithm for mutating genes in the **WebpageGenotype**
- **FitnessEvaluator**
  - Defined as a Strategy that determines the fitness of the genotype
    - The fitness evaluator iterates through the **GeneticComponents** in the Genotype to determine an overall fitness
    - It then determines a fitness level for each of the **Surveys** for the genotype to add to the overall fitness
- **Survey**
  - Is used in part of the fitness test, and represents a score applied after computing the genotypes fitness
  - Holds a reference to the Genotype that it applies to.
- **UIApplicationEngine**
  - Mediator that emulates the Evolutionary Generation of the webpages.
  - Uses Mating, Mutation, Fitness, Genotypes, and Phenotypes to cycle the Evolutionary Process.
- **UIDataPoint**
  - Keeps track of the time spent, as well as whether or not there was a "click" in a particular div (Genotype) of the given Phenotype (current webpage). This is the

primary data used in determining the fitness score for each of the genetic components

- **HTMLReport**
  - Contains a list of UIDataPoints, and inherits from the Report interface in the framework.
- **GeneticComponent**
  - A Composite structure that holds the HTML and CSS that is being examined.
    - **HTMLGeneticProperty**
      - A Composite that represents the tree of the HTML hierarchy.
    - **HTMLDecorator**
      - A Decorator that that represents the **HTMLClass** and **HTMLID** applied to a given HTML tags.
- **WebpageGenotype**
  - A Genotype of a web page representation.
  - Inherits from genotypetree in the framework which has a reference to the genetic component tree
- **DNA**
  - Inherits from Gene from the framework. Contains a list of HTML tags and a hashmap of the styles that are associated with each . It is accessed using **HTMLTag** as a key.
  - **HTMLTag**
    - A gene representation of DNA
  - **CSSStyle**
    - A gene representation of DNA
- **StyleProperty**
  - The data of the style, wrapped in a class.