МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ)

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ
И ОПТИКИ»
(СПбГУ ИТМО)

УТВЕРЖДАЮ
Ректор СПбГУ ИТМО,
докт. техн. наук, профессор
В. Н. Васильев

____ _____ 2008 г.

ПРОГРАММНОЕ СРЕДСТВО 3GENETIC

ПРОГРАММНЫЙ МОДУЛЬ COMMON
ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

7.190.00001-01 13 01-ЛУ

Инв.N подл.	Подп. и дата	Взам.инв.N	Инв.N дубл.	Подп. и дата

Декан факультета «Информационные
технологии и программирование»
докт. техн. наук, профессор
_____ В. Г. Парфенов

Руководитель темы
заведующий кафедрой «Технологии программирования»,
докт. техн. наук, профессор
_____ А. А. Шалыто

2008

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ)

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ
И ОПТИКИ»
(СПбГУ ИТМО)

УТВЕРЖДЕНО
7.190.00001-01 13 01-ЛУ

ПРОГРАММНОЕ СРЕДСТВО 3GENETIC

ПРОГРАММНЫЙ МОДУЛЬ COMMON
ОПИСАНИЕ ПРОГРАММЫ

7.190.00001-01 13 01

Листов 15

Инв. N подл.	Подп. и дата	Взам. инв. N	Инв. N дубл.	Подп. и дата

2008

АННОТАЦИЯ

В данном документе приводится описание модуля common средства *3Genetic*. Данный модуль является составной частью программного средства и представляет собой библиотеку интерфейсов.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Модуль common написан на языке программирования *Java*.
Для нормального функционирования данной программы необходимо, чтобы на персональной ЭВМ была установлена одна из следующих операционных систем:

- *Microsoft Windows 2000 Professional* (русская и английская версии), с установленным *Service Pack 4*;
- *Windows XP Professional* (русская и английская версии), с установленным *Service Pack 3*.

Также необходимо, чтобы на персональной ЭВМ была установлена среда разработки программного обеспечения на языке *Java*, версия не ниже *jdk1.6.0_xx*.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Библиотека интерфейсов программного средства 3GENETIC.

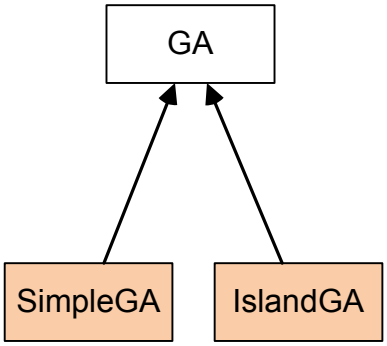
3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1.Интерфейс GA

3.1.1. Подробное описание

В этом интерфейсе описываются необходимые методы для модели генетического алгоритма (GA).

Граф наследования:



•

3.1.2. Открытые члены

- nextGeneration(): void
- getGeneration():List
- bigMutation():void
- getBest():Individual
- setGeneration(List<Individual>):void

3.1.3. Методы

- nextGeneration():void – генерирует следующее поколение;
- getGeneration():List – возвращает текущее поколение;
- bigMutation():void – производит «большую» мутацию;
- getBest():Individual – возвращает лучшую построенную особь;
- setGeneration(List<Individual> generation):void – задает поколение.
    - generation – новое поколение.

Объявления и описания членов интерфейса находятся в файле:

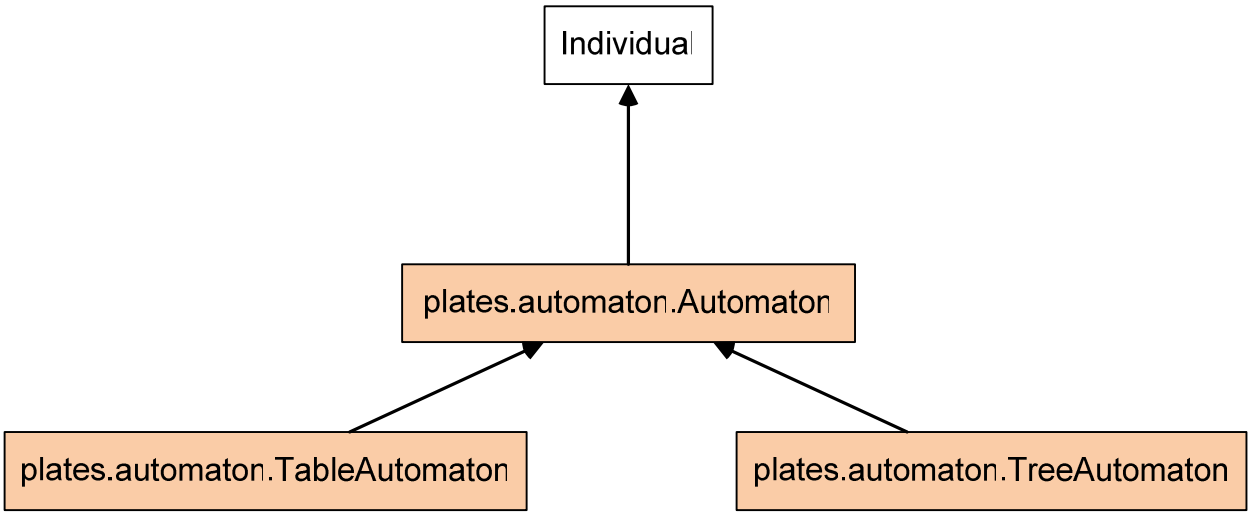
- \src\common\ga\GA.java

3.2.Интерфейс Individual

3.2.1. Подробное описание

В этом интерфейсе описаны необходимые методы для модели особи.

Граф наследования:



•

3.2.2. Открытые члены

- fitness():double
- mutate(Random):Individual
- crossover(Individual, Random):Individual[]

Методы

- fitness():double – возвращает значение функции приспособленности;
- mutate(Random r):Individual – возвращает результат мутации особи;
  - r – генератор случайных чисел.
- crossover(Individual p, Random r):Individual[] – возвращает результат скрещивания двух особей.
  - p – особь для скрещивания;
  - r – генератор случайных чисел.

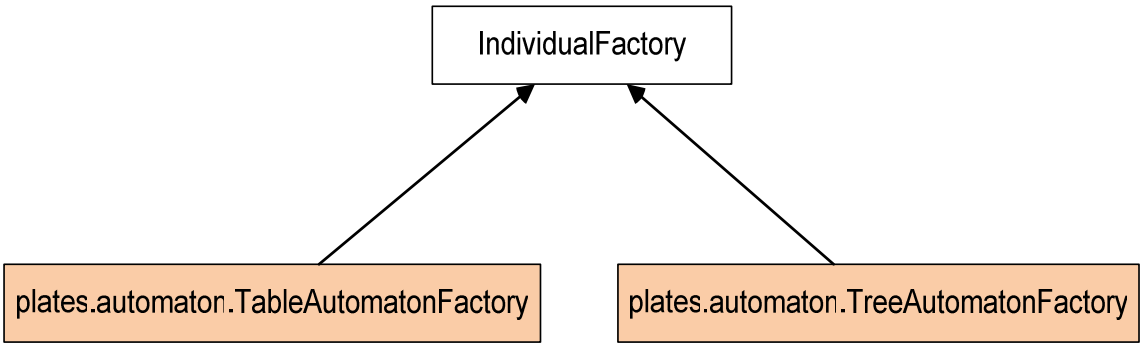
Объявления и описания членов класса находятся в файле:

- \src\common\ga\Individual.java

3.3.Интерфейс IndividualFactory

3.3.1. Подробное описание

В этом интерфейсе описываются необходимые методы для класса, генерирующего особи случайным образом.
Граф наследования:



3.3.2. Открытые члены

- randomIndividual():Individual

3.3.3. Методы

- randomIndividual():Individual – возвращает случайно сгенерированную особь.

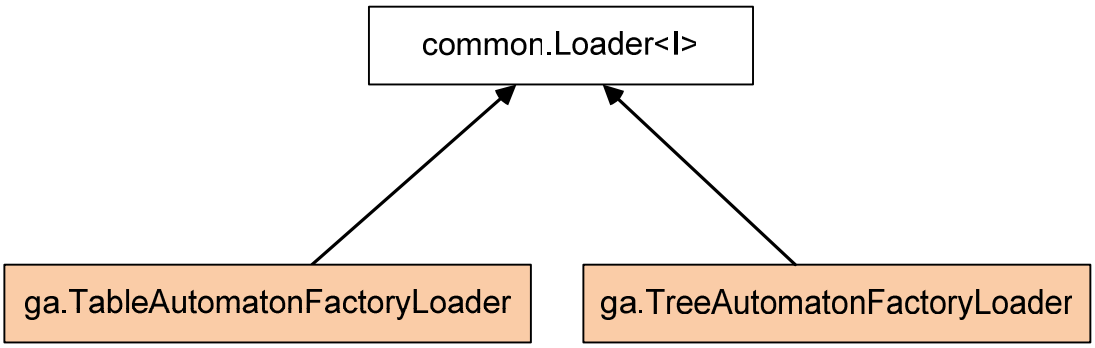
Объявления и описания членов класса находятся в файле:

- \src\common\ga\IndividualFactory.java

3.4.Интерфейс Loader<I>

3.4.1. Подробное описание

В этом интерфейсе описываются необходимые методы для создания объектов загруженных из jar-архивов, генерирующего особи случайным образом. Загружаемый класс назовем I.
Граф наследования:



3.4.2. Открытые члены

- load():<I>
- getProperties():Properties

3.4.3. Методы

- load():I – созданный экземпляр класса I;

- getProperties():Properties – возвращает набор настраеваемых параметров.

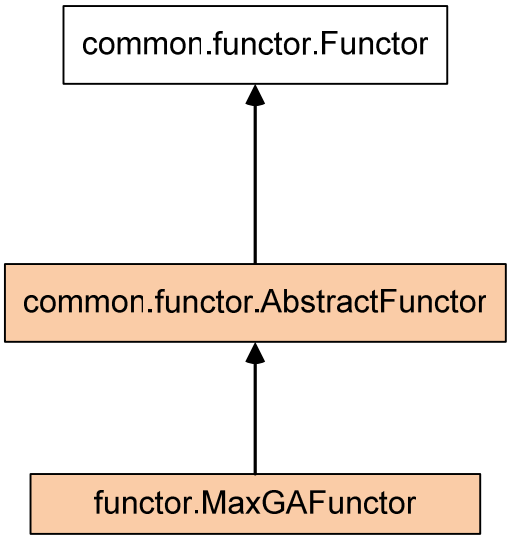
Объявления и описания членов класса находятся в файле:

- \src\common\ga\Loader.java

3.5.Интерфейс Functor

3.5.1. Подробное описание

В этом интерфейсе описываются необходимые методы для создания объектов загруженных из jar-архивов, генерирующего особи случайным образом. Загружаемый класс назовем I.
Граф наследования:



3.5.2. Открытые члены

- getValues():List<Double>
- update(List<Individual> generation):void
- clear():void

3.5.3. Методы

- getValues():List<Double> – возвращает значения для уже подсчитанных поколений;
- update(List<Individual> generation):void – добавление нового поколения;
  - generation – добавляемое поколение.
- clear():void – удаление всех предыдущих поколений.

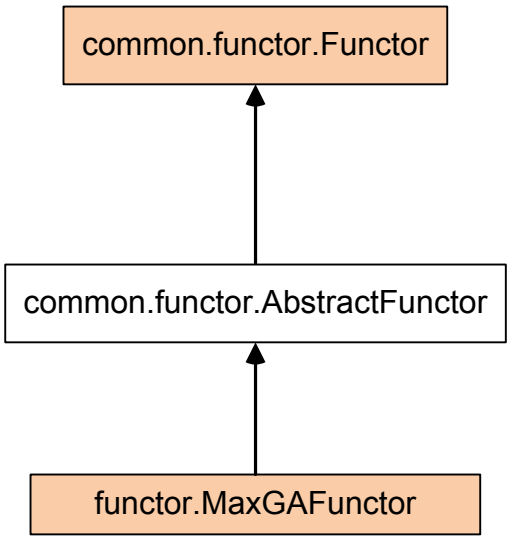
Объявления и описания членов класса находятся в файле:

- \src\common\ga\Functor.java
  -

3.6.Класс AbstractFunctor

Этот класс частично реализует интерфейс Functor.

Граф наследования:



Защищенные члены

- values: List<Double>.

3.6.1. Открытые члены

- getValues():List<Double>
- update(List<Individual> generation):void
- clear():void

3.6.2. Конструктор(ы)

- MaxGAFunctor() – создание нового функтора.

3.6.3. Методы

- getValues():List<Double> – возвращает значения для уже подсчитанных поколений;
- update(List<Individual> generation):void – добавление нового поколения;
  - generation – добавляемое поколение.
- clear() – удаление всех предыдущих поколений.

Объявления и описания членов класса находятся в файле:

- common\functor\AbstractFunctor.java
  -

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Для нормального функционирования программы автоматического построения с помощью генетического программирования конечных автоматов, управляющих системами со сложным поведением, на персональной ЭВМ, необходимо, чтобы аппаратное обеспечение персональной ЭВМ удовлетворяло следующим требованиям:

- процессор *Intel Pentium IV* или совместимый;
- тактовая частота процессора 2ГГц, не менее;
- оперативная память 1024 MB, не менее;
- дисковый накопитель объемом 80 GB, не менее;
- отображающее устройство (монитор) с поддержкой разрешения 1024x768;
- устройства ввода клавиатура и мышь (трекбол, тачпад);

5. ВЫЗОВ И ЗАГРУЗКА

public GA.nextGeneration(): void
public GA.getGeneration():List
public GA.bigMutation():void
public GA.getBest():Individual
public GA.setGeneration(List<Individual>):void

public Individual.fitness():double
public Individual.mutate(Random):Individual
public Individual.crossover(Individual, Random):Individual[]

public IndividualFactory.randomIndividual():Individual

public Loader<I>.load():<I>
public Loader<I>.getProperties():Properties

public Functor.getValues():List<Double>
public Functor.update(List<Individual> generation):void
public Functor.clear():void

public AbstractFunctor.AbstractFunctor()
public AbstractFunctor.getValues():List<Double>
public AbstractFunctor.update(List<Individual> generation):void
public AbstractFunctor.clear():void

6. ВХОДНЫЕ ДАННЫЕ

- GA.nextGeneration():void
  - нет.
- GA.getGeneration():List
  - нет.
- GA.bigMutation():void
  - нет.
- GA.getBest():Individual
  - нет.
- GA.setGeneration(List<Individual> generation):void
  - generation – новое поколение.

- Individual.fitness():double
  - нет
- Individual.mutate(Random r):Individual
  - r – генератор случайных чисел.
- Individual.crossover(Individual p, Random r):Individual[]
  - p – особь для скрещивания;
  - r – генератор случайных чисел.

- IndividualFactory.randomIndividual():Individual
  - нет

- Loader<I>.load():I
  - нет
- Loader<I>.getProperties():Properties
  - нет

- Functor.getValues():List<Double>
  - нет
- Functor.update(List<Individual> generation):void
  - generation – добавляемое поколение.
- Functor.clear():void
  - нет

7. ВЫХОДНЫЕ ДАННЫЕ

- GA.nextGeneration():void – нет;
- GA.getGeneration():List<Individual> – возвращает текущее поколение;
- GA.bigMutation():void – нет;
- GA.getBest():Individual – возвращает лучшую построенную особь;
- GA.setGeneration(List<Individual> generation):void – нет.

- Individual.fitness():double – возвращает значение функции приспособленности;
- Individual.mutate(Random r):Individual – возвращает результат мутации особи;
- Individual.crossover(Individual p, Random r):Individual[] – возвращает результат скрещивания двух особей.

- IndividualFactory.randomIndividual():Individual – возвращает случайно сгенерированную особь.

- Loader<I>.load():I – созданный экземпляр класса I;
- Loader<I>.getProperties():Properties – возвращает набор настраеваемых параметров.

- Functor.getValues():List<Double> – возвращает значения для уже подсчитанных поколений;
- Functor.update(List<Individual> generation):void – нет;
- Functor.clear():void – нет.

- AbstractFunctor.getValues():List<Double> – возвращает значения для уже подсчитанных поколений;
- AbstractFunctor.update(List<Individual> generation):void – нет;

AbstractFunctor.clear():void – нет.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ докумен-та*	*Входящий № сопрово-дительно го докум. и дата*	*Подп.*	*Дата*
Изм.	*Измененных*	*Замененных*	*Новых*	*Аннулированных*					