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 PLATE
ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

7.190.00001-01 13 06-ЛУ

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

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

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 06-ЛУ

ПРОГРАММНОЕ СРЕДСТВО 3GENETIC

ПРОГРАММНЫЙ МОДУЛЬ PLATE
ОПИСАНИЕ ПРОГРАММЫ

7.190.00001-01 13 06

Листов 44

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

2008

АННОТАЦИЯ

В данном документе приводится текст модуля plate программного средства 3GENETIC, содержащего реализацию особей генетического программирования для задачи построения управляющего автомата для беспилотных летательных аппаратов.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Программа построения автоматов управления системами со сложным поведением с помощью генетического программирования, написана 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

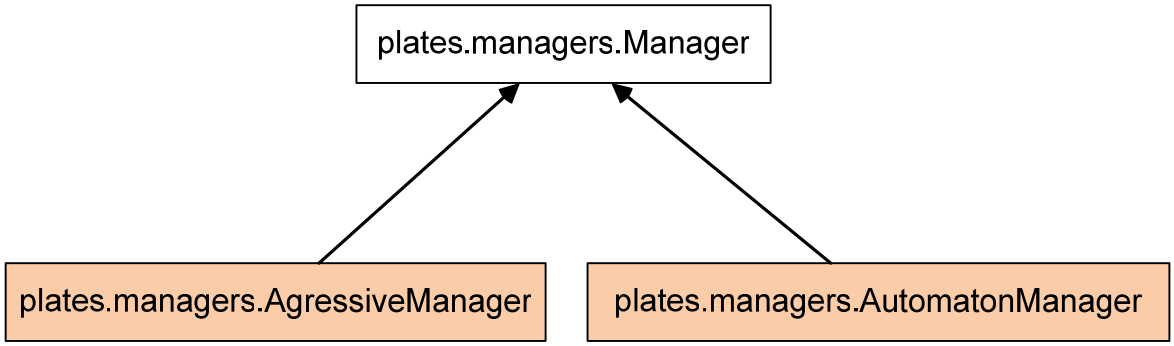
Данный модуль является подключаемым (то есть программное средство 3Genetic способно работать в его отсутствии). Расширяет возможности ярда 3Genetic. Содержит реализацию особей генетического программирования для задачи о построении управляющего автомата для бесплотных летательных аппаратов.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1.Интерфейс Manager

3.1.1. Подробное описание

В этом интерфейсе описываются необходимые методы для класса, управляющего моделью беспилотного летательного аппарата.
Граф наследования:



3.1.2. Открытые члены

- doTurn():void
- getPlates():List
- init(List, List):void

3.1.3. Методы

- doTurn():void – отвечает за выработку выходных воздействий на модель беспилотного летательного аппарата;
- getPlates():List – возвращает список моделей беспилотных летательных аппаратов;
- init(List<Plate> plates, List<Plate> allPlates):void – задает списки моделей беспилотных летательных аппаратов.
    - plates – список беспилотных летательных объектов, являющихся сокомандниками;
    - allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

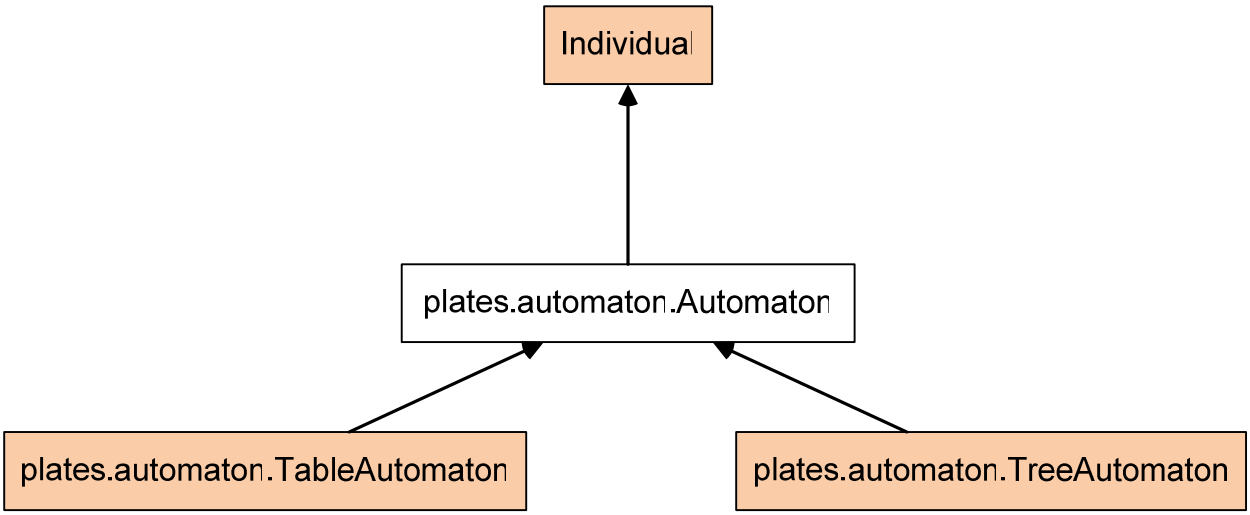
Объявления и описания членов класса находятся в файле:

- \src\plates\managers\Manager.java

3.2.Класс Automaton

3.2.1. Подробное описание

Класс реализует связь между моделированием соревнований и представлениями автоматов.
Граф наследования:



3.2.2. Защищенные члены

- plate:Plate – модель беспилотного летательного аппарата;
- fitness:double – значение функции приспособленности данного автомата;
- getPlate():Plate – возвращает модель беспилотного летательного аппарата, которой осуществляется управление;

3.2.3. Открытые члены

- Automaton()
- fitness():double
- compareTo(Individual):int
- doTurn(boolean[]):void
- repairedAutomaton():Automaton
- clone():Automaton
- setPlate(Plate):void

3.2.4. Методы

- fitness():double – возвращает значение функции приспособленности;
- compareTo(Individual o):int – сравнивает значение функции приспособленности для со значением функции приспособленности другой особи;
- – сравниваемая особь
- getPlate():Plate – возвращает модель беспилотного летательного аппарата, которой осуществляется управление;
- setPlate(Plate plate):void – задает модель беспилотного летательного аппарата, которой будет осуществляться управление.
    - plate – модель беспилотного летательного аппарата, которой будет осуществляться управление

Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\Automaton.java

3.3.Класс Fitness

3.3.1. Подробное описание

Класс, вычисляющий значение функции приспособленности для особи путем проведения соревнований с заданными системами управления моделью беспилотного летательного аппарата.
Наследников нет.

3.3.2. Защищенные члены

- readOpponent(String file):Manager – прочитать описание системы управления моделью беспилотного летательного аппарата из файла;
- opponent1:Manager – первый противник;
- opponent2:Manager – второй противник.

3.3.3. Открытые члены

- fitness(Automaton):double
- fitness2(Automaton):double

1.

3.3.4. Конструктор(ы)

Fitness()

3.3.5. Методы

- fitness(Automaton a):double – подсчет значения функции приспособленности в десяти соревнованиях;
  - a – автомат у которого подсчитывается функция приспособленности
- fitness2(Automaton a):double – подсчет значения функции приспособленности в тридцати соревнованиях;
  - a – автомат у которого подсчитывается функция приспособленности

Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\Fitness.java

3.4.Класс State

3.4.1. Подробное описание

Класс, представляющий собой состояние автомата. Хранит информацию о переходах и действиях с использованием метода сокращенных таблиц переходов.
Классы наследники: отсутствуют.

3.4.2. Защищенные члены

- State()
- endState:int[] – массив номеров состояний, в которые ведут переходы из данного состояния;
- transitionTable:boolean[][] – массив списков действий, выполняемых при переходе из этого состояния, в зависимости от значений значимых переменных;
- variable:boolean[] – массив значимых переменных.

3.4.3. Открытые члены

- State(int[], boolean[][], boolean[])
- mutate(Random):State
- crossover(State, Random):State[]
- choosePred(State, State, State, Random):void
- setEndState(int, int):State
- getEndState(int):int
- getActions(int):boolean[]
- getVariable(int):int
- clone():State

3.4.4. Конструктор(ы)

- State(int[] endState, boolean[][] transitionTable, boolean[] variable) – создает новое состояние;
  - endState – массив номеров конечных состояний;
  - transitionTable – матрица переходов;
  - variable – массив значимых переменных.
- State() – создает «пустое» состояние.

3.4.5. Методы

- mutate(Random r):State – возвращает состояние после применения к нему операции мутации;
  - r – генератор случайных чисел.
- crossover(State p, Random r):State[] – возвращает два состояния, получаемые в результате скрещивания данного и передаваемого методу в качестве параметра;
  - p – второе состояние для скрещивания;
  - r – генератор случайеых чисел.
- choosePred(State par, State ch1, State ch2, Random r):void – выполняет выбор значимых переменных “детей” данного состояния. Применяется при выполнении операции скрещивания;
  - par – один из “родителей”;
  - ch1 – один из “детей”;
  - ch2 – один из “детей”;
  - r – генератор случайных чисел.

- setEndState(int i, int en):State – устанавливает номер состояния, в которое ведет переход из данного состояния при заданной комбинации значений значимых переменных;
    - i – номер комбинации входных переменных;
    - en – номер конечного состояния.
- getEndState(int i):int – возвращает номер состояния, в которое ведет переход из данного состояния, по комбинации значений значимых переменных;
    - i – номер комбинации входных переменных.
- getActions(int i):boolean[] – возвращает список действий, которые выполняются при выборе перехода из данного состояния, по комбинации значений значимых переменных;
    - i – номер комбинации входных переменных.
- getVariable(int i):int – возвращает является ли данная переменная значимой;
    - i – номер опращиваемой переменной.
- clone():State – возвращает копию данного состояния.

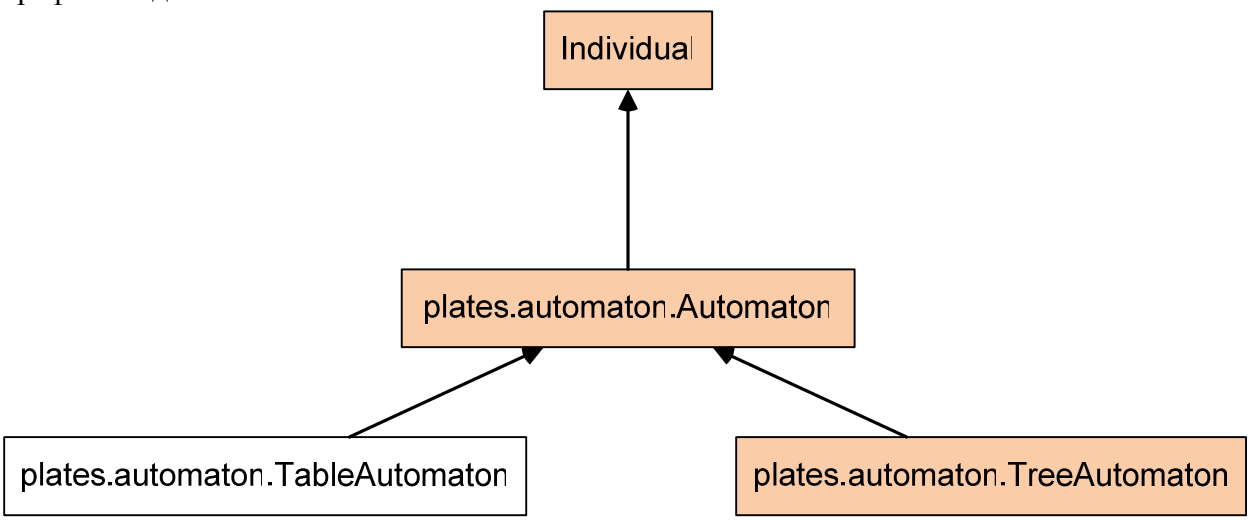
Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TableAutomaton.java

3.5.Класс TableAutomaton

3.5.1. Подробное описание

Класс, представляющий автомат с помощью метода сокращенных таблиц переходов. Хранит таблицу переходов для каждого состояния. Содержит реализацию операций мутации и скрещивания.
Граф наследования:



3.5.2. Защищенные члены

- toIntString(int[]):String
- toBoolString(boolean[]):String
- numberVariables:int – число значимых переменных;
- numberActions:int – число действий;
- numberStates:int – число состояний автомата;
- initialState:int – номер начального состояния;
- currentState:int – текущее состояние;

- countAllVariables:int – общее число переменных;
- state:State[] – массив состояний;
- mark:boolean[] – массив флагов, используемый при работе алгоритма восстановления связей между состояниями автомата;
- toBoolString(boolean[]):String – преобразует массив логических переменных в строку;
- toIntString(int[]):String – преобразует массив целых чисел в строку.

3.5.3. Открытые члены

- TableAutomaton(int, int, int, int, int)
- getNumberVariables():int
- getNumberStates():int
- getInitialState():int
- mutate(Random):TableAutomaton
- crossover(Individual, Random):TableAutomaton[]
- doTurn(boolean[]):void
- setState(int, State):void
- getState(int):State
- repairedAutomaton():TableAutomaton
- clone():Automaton
- toString():String

2.

3.5.4. Статические открытые данные

- ENDSTATEMUTATION:double = 0.3 – вероятность изменения номера состояния, в которое ведет переход, при выполнении операции мутации;
- ACTIONSMUTATION:double = 0.3 – вероятность изменения действия, выполняемого при выборе перехода, при выполнении операции мутации;
- PREDICATMUTATION:double = 0.3 – вероятность изменения набора значимых переменных при выполнении операции мутации.

3.5.5. Конструктор(ы)

- TableAutomaton(int numberStates, int numberActions, int numberVariables, int countAllVariables, int is) – создает автомат с заданным числом состояний, числом действий, числом значимых переменных, числом переменных и номером начального состояния.
  - numberStates – число состояний автомата;
  - numberActions – число возможных вывоздных действий;
  - numberVariables – число знамых переменных;
  - countAllVariables – число возможный входных переменных;
  - is – номер стартового состояния.

3.5.6. Методы

- getNumberVariables():int – возвращает число переменных;
- getNumberStates():int – возвращает число состояний;
- getInitialState():int – возвращает номер начального состояния;
- mutate(Random r):TableAutomaton – возвращает автомат, к которому применена операция мутации;
    - r – генератор случайных чисел;
- crossover(Individual p, Random r):TableAutomaton[] – возвращает два автомата, получаемых в результате применения операции скрещивания к данному и переданному автоматам в качестве параметра;
    - p – особь для скрещивания;
    - r – генератор случайных чисел;
- doTurn(boolean[] variables):void – выполняет переход в очередное состояние и выработку выходных воздействий;
    - variables – массив значений входных переменных;
- setState(int i, State s):void – заменяет состояние на переданное в качестве параметра;
    - i – номер заменяемого состояния;
    - s – новое состояние;
- getState(int i):State – возвращает состояние по номеру;
    - i – номер состояния;
- repairedAutomaton():TableAutomaton – применяет к автомату алгоритм восстановления связей;
- clone():Automaton – возвращает копию автомата;
- toString():String – возвращает строковое представление автомата.

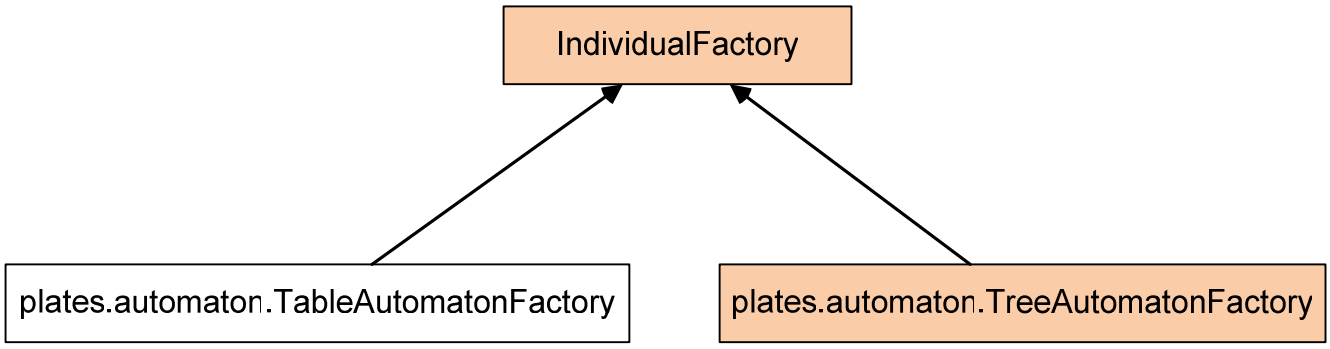
Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TableAutomaton.java

3.6.Класс TableAutomatonFactory

3.6.1. Подробное описание

Класс, генерирующий случайным образом автоматы, представленные с помощью сокращенных таблиц переходов.
Граф наследования:



3.6.2. Защищенные члены

- numberStates:int – число состояний;

- countAllVariables:int – число переменных;
- numberVariables:int – число значимых переменных;
- numberActions:int – число выходных воздействий;
- pAction:double – вероятность, с которой будет выполнено действие;
- RANDOM:Random = new Random() – генератор случайных чисел.

3.6.3. Открытые члены

- TableAutomatonFactory(int numberStates, int countAllVariables, int numberVariables, int numberActions, double pAction)
- randomIndividual():TableAutomaton

3.6.4. Конструктор(ы)

- TableAutomatonFactory(int numberStates, int countAllVariables, int numberVariables, int numberActions, double pAction) – создает генератор автоматов с заданным числом состояний, числом действий, числом значимых переменных, числом переменных и номером начального состояния и заданной вероятностью появления каждого выходного воздействия.
    - numberStates – число состояний автомата;
    - numberActions – число возможных вывоздных действий;
    - numberVariables – число знамых переменных;
    - countAllVariables – число возможный входных переменных;
    - pAction – вероятность действия на переходе.

3.6.5. Методы

- randomIndividual():TableAutomaton – возвращает случайно сгенерированный автомат в представлении с помощью сокращенных таблиц переходов.

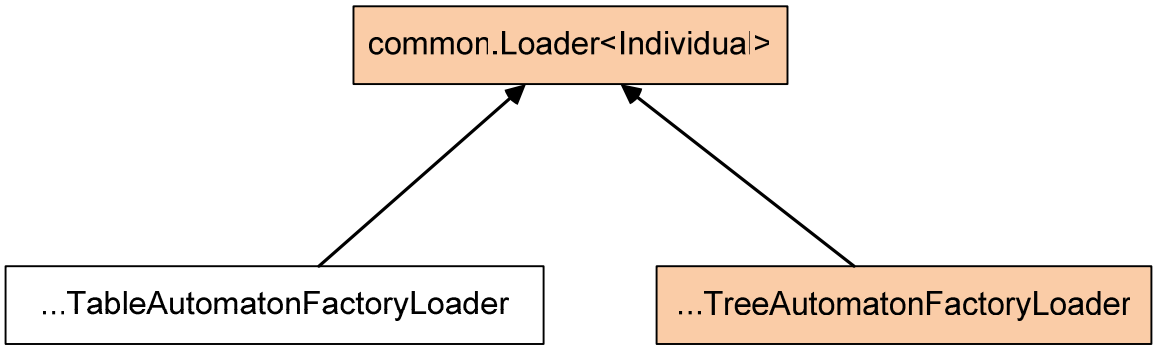
Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TableAutomatonFactory.java

3.7.Класс TableAutomatonFactoryLoader

3.7.1. Подробное описание

Класс, создающий фабрику автоматов представленных в виде сокращенных таблиц переходов.
Граф наследования:



3.7.2. Открытые члены

- TableAutomatonFactoryLoader(JarFile jr)
- load():TableAutomatonFactory
- getProperties():Properties

3.7.3. Конструктор(ы)

- TableAutomatonFactoryLoader(JarFile jr) – создает загрузчик фабрики автоматов.
  - jr – jar-архив, в котором находится фабрика;

3.7.4. Методы

- load():TableAutomatonFactory – возвращает новую фабрику автоматов;
- getProperties():Properties – возвращает список настраиваемых параметров алгоритма;

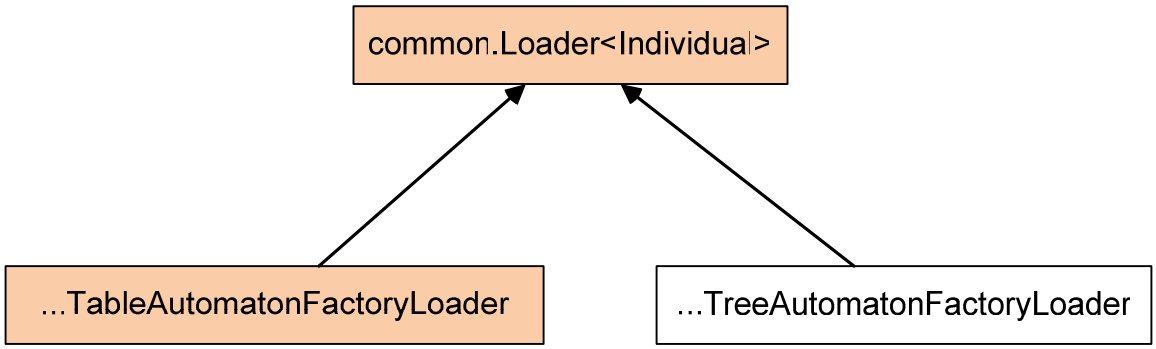
Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TableAutomatonFactoryLoader.java

3.8.Класс TreeAutomatonFactoryLoader

3.8.1. Подробное описание

Класс, создающий фабрику автоматов представленных в виде сокращенных таблиц переходов.
Граф наследования:



3.8.2. Открытые члены

- TreeAutomatonFactoryLoader(JarFile jr)
- load():TreeAutomatonFactory
- getProperties():Properties

3.8.3. Конструктор(ы)

- TreeAutomatonFactoryLoader(JarFile jr) – создает загрузчик фабрики автоматов.
    - jr – jar-архив, в котором находится фабрика;

3.8.4. Методы

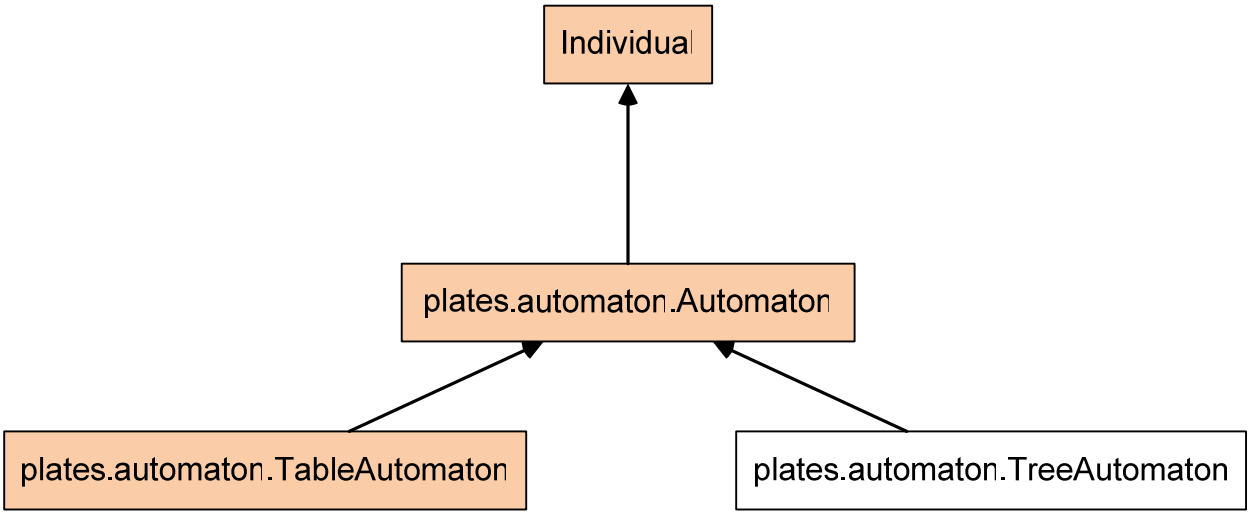
- load():TreeAutomatonFactory – возвращает новую фабрику автоматов;
- getProperties():Properties – возвращает список настраиваемых параметров алгоритма;

Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TreeAutomatonFactoryLoader.java
    -

3.9.Класс TreeAutomaton

Класс, реализующий автомат, представленный с помощью деревьев решений.
Граф наследования:



3.9.1. Защищенные члены

- state:Tree[] – массив состояний, для каждого из которого хранится свое дерево решений;
- initialState:int – номер начального состояния автомата;
- currentState:int – номер текущего состояния автомата;

- height:int – рекомендуемая глубина дерева решений для состояний автомата;
- fitness:double – значение функции приспособленности автомата;
- fact:TreeAutomatonFactory – генератор автоматов, представленных в виде деревьев решений.

3.9.2. Открытые члены

- TreeAutomaton(int, int, int, TreeAutomatonFactory)
- doTurn(boolean[]):void
- repairedAutomaton():TreeAutomaton
- setState(int, Tree):void
- getState(int):Tree
- getInitialState():int
- getNumberStates():int
- getCountAllVariables():int
- getNumberActions():int
- getHeight():int
- clone():TreeAutomaton
- mutate(Random):TreeAutomaton
- crossover(Individual, Random):TreeAutomaton[]
- fitness():double
- toString():String

3.

3.9.3. Статические открытые данные

- VERTEX_MUTATION_PROBABILITY:double = 0.5 – вероятность мутации узла дерева;
- VERTEX_CROSSOVER_PROBABILITY:double = 0.5 – вероятность скрещивания для узла дерева решений;
- PENALTY:int = 3 – множитель в функции приспособленности, отвечающий за превышение рекомендуемой высоты дерева решений.

3.9.4. Конструктор(ы)

- TreeAutomaton(int numberStates, int numberVariables, int number Actions, TreeAutomatonFactory factory) – создает автомат.
    - numberStates – число состояний автомата;
    - numberActions – число возможных вывоздных действий;
    - numberVariables – число знамых переменных;
    - factory – генератор случайных деревьев решений;

3.9.5. Методы

- doTurn(boolean[] variables):void – выполняет переход в следующее состояние и выработу выходных воздействий;
- setState(int i, Tree t):void – заменяет состояние на переданное в качестве параметра;

  - i – номер заменяемого состояния;
  - t – новое состояние(дерево решений);
- getState(int i):Tree – возвращает состояние по номеру;
  - i – номер состояния;
- getInitialState():int – возвращает номер начального состояния;
- getNumberStates():int – возвращает число состояний;
- getCountAllVariables():int – возвращает число переменных;
- getNumberActions():int – возвращает число действий;
- getHeight():int – возвращает рекомендуемую высоту дерева решений;
- clone():TreeAutomaton – возвращает копию автомата;
- mutate(Random r):TreeAutomaton – возвращает автомат, к которому применена операция мутации;
  - r – генератор случайных чисел;
- crossover(Individual p, Random r):TreeAutomaton[] – возвращает результат применения операции скрещивания к данному автомату и автомату, переданному в качестве параметра;
  - p – особь для скрещивания;
  - r – генератор случайных чисел;
- fitness():double – возвращает значение функции приспособленности;
- toString():String – возвращает строковое представление автомата.

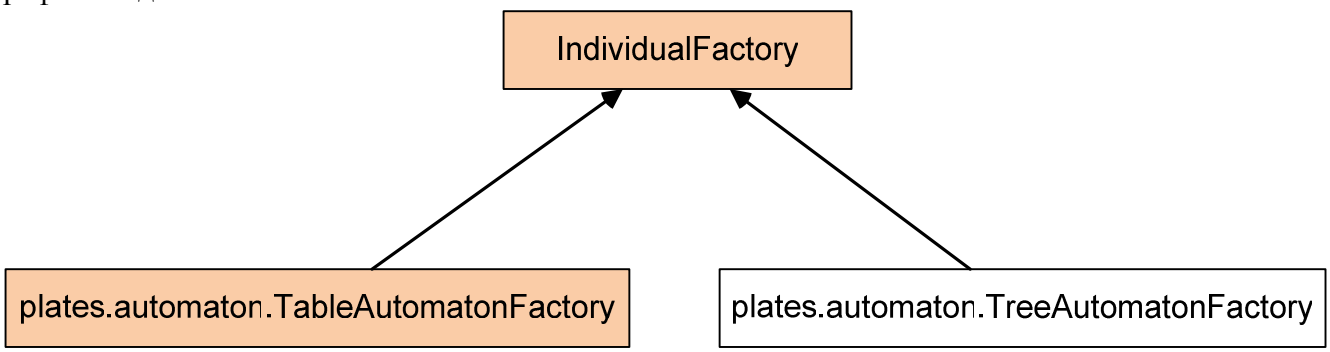
Объявления и описания членов класса находятся в файле:

- \src\plates\automaton\TreeAutomaton.java

3.10. Класс TreeAutomatonFactory

3.10.1. Подробное описание

Класс, генерирующий случайным образом автоматы, представленные с помощью деревьев решений.
Граф наследования:



3.10.2. Защищенные члены

- numberStates:int – число состояний автоматов;
- countAllVariables:int – число переменных;
- numberActions:int – число выходных воздействий;
- pAct:double – вероятность появления заданного выходного воздействия;
- height:int – рекомендуемая высота дерева решений;
- VERTEXPROBABILITY:double = 0.75 – вероятность того, что текущая генерируемая вершина внутренняя (используется при генерации случайного дерева решений);

- RANDOM:Random = new Random() – генератор случайных чисел.

3.10.3. Открытые члены

- TreeAutomatonFactory(int, int, int, double, int)
- getCountAllVariables():int
- getNumberActions():int
- randomIndividual():TreeAutomaton
- randomTree(int):TreeNode

4.

3.10.4. Статические открытые данные

Отсутствуют.

3.10.5. Конструктор(ы)

- TreeAutomatonFactory(int numberStates, int countAllVariables, int numberActions, double pAct, int height) – задает число состояний, число переменных, число выходных воздействий, вероятность появления каждого выходного воздействия и рекомендуемую высоту дерева решений генерируемых автоматов.
    - numberStates – число состояний автомата;
    - numberActions – число возможных вывоздных действий;
    - numberVariables – число знамых переменных;
    - pAct – вероятность действия на переходе;
    - height - рекоммендуемая высота дерева решений.

3.10.6. Методы

- getCountAllVariables():int – возвращает число переменных;
- getNumberActions():int – возвращает число выходных воздействий;
- randomIndividual():TreeAutomaton – возвращает случайно сгенерированный автомат, представленный с помощью деревьев решений;
- randomTree(int h):TreeNode – возвращает случайно сгенерированное дерево решений;
    - h - высота уже сгенерированной части

Объявления и описания членов класса находятся в файле:

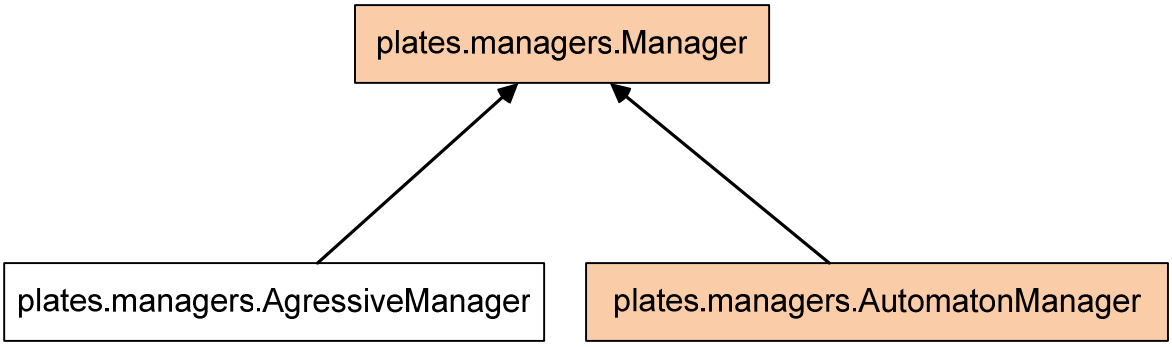
- \src\plates\automaton\TreeAutomatonFactory.java

3.11. Класс AggressiveManager

3.11.1. Подробное описание

Класс, реализующий «агрессивную» стратегию управления моделью беспилотного летательного аппарата.

Граф наследования:



3.11.2. Защищенные члены

- plates:List – список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
- random:Random = new Random() – генератор случайных чисел.

3.11.3. Открытые члены

- doTurn():void
- getPlates():List
- init(List, List):void

3.11.4. Конструктор(ы)

- AgressiveManager()

3.11.5. Методы

• doTurn():void – вырабатывает очередные выходные воздействия на управляемые модели беспилотных летательных аппаратов;
• getPlates():List – возвращает список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
• init(List<Plate> plates, List<Plate> allPlates):void – установить список моделей беспилотных летательных аппаратов, которыми осуществляется управление.

- plates – список беспилотных летательных объектов, являющихся сокомандниками;
- allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

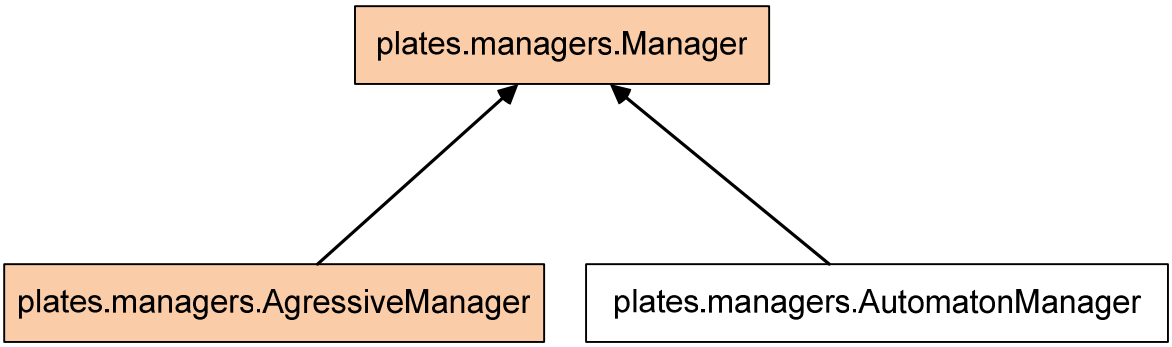
Объявления и описания членов класса находятся в файле:

- \src\plates\managers\AggressiveManager.java

3.12. Класс AutomatonManager

3.12.1. Подробное описание

Класс, управляющий моделями беспилотных летательных аппаратов с помощью автоматов. Вычисляет значения входных переменных по положению и состоянию всех беспилотных летательных аппаратов, участвующих в соревновании.
Граф наследования:



3.12.2. Защищенные члены

- canHit(Plate, Plate):Boolean – определяет, может ли один беспилотный летательный аппарат столкнуться с другим, если они будут двигаться равномерно и прямолинейно с текущего момента времени;
- automata:List – список автоматов, которым передаются вычисленные значения входных переменных;
- plates:List – список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
- allPlates:List – список всех моделей беспилотных летательных аппаратов.

3.12.3. Открытые члены

- AutomatonManager(Automaton)
- doTurn():void
- getPlates():List
- init(List, List):void

5.

3.12.4. Статические открытые данные

Отсутствуют.

3.12.5. Конструктор(ы)

- AutomatonManager(Automaton a) – задает автоматы для всех моделей беспилотных летательных аппаратов.
  - a – управляющий автомат.

3.12.6. Методы

- doTurn():void – передает вычисленные значения входных переменных автоматам, которые, в свою очередь, вырабатывают выходные воздействия на модели беспилотных летательных аппаратов;
- getPlates():List – возвращает список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
- init(List<Plate> plates, List<Plate> allPlates):void – задать список моделей беспилотных летательных аппаратов.
    - plates – список беспилотных летательных объектов, являющихся сокомандниками;
    - allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

Объявления и описания членов класса находятся в файле:

- \src\plates\managers\AutomatonManager.java

3.13. Класс Vector

3.13.1. Подробное описание

Хранит двумерный вектор.

3.13.2. Открытые члены

- Vector(double, double)
- getLength():double
- multiply(double):Vector
- multiply(Vector):double
- add(Vector):Vector
- subtract(Vector):Vector
- rotate(double):Vector
- x:double – абсцисса;
- y:double – ордината.

3.13.3. Конструктор(ы)

- Vector(double x, double y) – создает вектор с координатами, переданными в качестве параметров.
    - x – абсцисса;
    - y – ордината.

3.13.4. Методы

- getLength():double – возвращает длину вектора;
- multiply(double k):Vector – умножает вектор на число;
    - k – множитель.

- multiply(Vector v):double – возвращает скалярное произведение с вектором, переданным в качестве параметра;
    - v – множитель.
- add(Vector v):Vector – прибавляет вектор;
    - v – слагаемое.
- subtract(Vector v):Vector – вычитает вектор;
    - v - вычитаемое
- rotate(double phi):Vector – поворачивает вектор на угол.
    - phi – угол поворота

Объявления и описания членов класса находятся в файле:

- \src\plates\utils\Vector.java

3.14. Класс Competition

3.14.1. Подробное описание

Этот класс реализует соревнование моделей беспилотных летательных аппаратов.
Классы наследники: отсутствуют.

3.14.2. Защищенные члены

- genRandomCoordinates(int):double[]
- emulate():void
- result:double[] = {Double.NEGATIVE_INFINITY, ... – результаты соревнования;
- manager:Manager[] –система управления моделями беспилотных летательных аппаратов;
- logic:GameLogic – реализация логики соревнования;
- randomXCoordinates:double[] – координаты моделей беспилотных летательных аппаратов.

3.14.3. Открытые члены

- Competition(Manager, Manager)
- getResult(int):double

3.14.4. Конструктор(ы)

- Competition(Manager man1, Manager man2) – задает системы управления моделями беспилотных летательных аппаратов для двух команд;
    - man1 – тестируемая команда.
    - man2 – соперник.

3.14.5. Методы

- genRandomCoordinates(int index):double[] – случайным образом расставляет модели беспилотных летательных аппаратов;
    - index – номер модели.

- getResult(int index):double – возвращает результат соревнования;
    - index – номер команды.
- emulate():void – запускает эмулятор соревнования моделей беспилотных летательных аппаратов.

Объявления и описания членов класса находятся в файле:

- \src\plates\Competition.java

3.15. Класс Config

3.15.1. Подробное описание

Класс, содержащий настройки.
Классы наследники: отсутствуют.

3.15.2. Открытые члены

- getInfluenceDistance():double
- getTimeStep():double
- getConstantT():double
- getConstantF1():double
- getConstantF2():double
- getFieldHeight():int
- getFieldWidth():int
- getPlateDiameter():int
- getPlatesCount():int
- getMaximalRotateAngle():int
- getInitFuel():double
- getInitSpeed():double

6.

3.15.3. Методы

- getInfluenceDistance():double – возвращает максимальное расстояние, на котором проявляется аэродинамическое взаимодействие;
- getTimeStep():double – возвращает шаг по времени;
- getConstantT():double – возвращает константу *T*;
- getConstantF1():double – возвращает константу *F1*;
- getConstantF2():double – возвращает константу *F2*;
- getFieldHeight():int – возвращает высоту поля;
- getFieldWidth():int – возвращает ширину поля;
- getPlateDiameter():int – возвращает диаметр модели беспилотного летательного аппарата;
- getPlatesCount():int – возвращает число моделей беспилотных летательных аппаратов в каждой команде;
- getMaximalRotateAngle():int – возвращает максимальный угол поворота;
- getInitFuel():double – возвращает начальное количество топлива;
- getInitSpeed():double – возвращает начальную скорость.

Объявления и описания членов класса находятся в файле:

- \src\plates\Config.java

3.16. Класс GameLogic

3.16.1. Подробное описание

Класс, описывающий логику игры.
Классы наследники: отсутствуют.

3.16.2. Защищенные члены

- EPS:double = 1e-9 – погрешность вычислений;
- plates:List – список моделей беспилотных летательных аппаратов.

3.16.3. Открытые члены

- GameLogic(List)
- processSlowPlates():void
- calculateNewSpeeds():void
- movePlates():void
- processFlyingOutPlates():void
- gameOver():boolean

3.16.4. Конструктор(ы)

- GameLogic(List<Plates> plates) – задает участвующие в соревновании модели беспилотных летательных аппаратов.
    - plates – участвующие модели.

3.16.5. Методы

- processSlowPlates():void – обрабатывает модели беспилотных летательных аппаратов, двигающиеся со скоростью менее одного метра в секунду;
- calculateNewSpeeds():void – пересчитывает скорости моделей беспилотных летательных аппаратов;
- movePlates():void – перемещает модели беспилотных летательных аппаратов;
- processFlyingOutPlates():void – обрабатывает модели беспилотных летательных аппаратов, вышедшие за границы трассы соревнований;
- gameOver():boolean – проверяет, закончилось ли соревнование.

Объявления и описания членов класса находятся в файле:

- \src\plates\GameLogic.java

3.17. Класс Plate

3.17.1. Подробное описание

Класс, описывающий модель беспилотного летательного аппарата.
Классы наследники: отсутствуют.

3.17.2. Защищенные члены

- state:State – состояние модели беспилотного летательного аппарата (полет, посадка, авария);
- position:Vector – вектор, задающий координаты модели беспилотного летательного аппарата;
- speed:Vector – вектор скорости модели беспилотного летательного аппарата;
- fuel:double – остаток топлива;
- q:double – текущий расход топлива;
- a:double – угол поворота.

3.17.3. Открытые члены

- Plate(Vector, Vector, double)
- getPosition():Vector
- setPosition(Vector):void
- getSpeed():Vector
- setSpeed(Vector):void
- getFuel():double
- getQ():double
- setQ(double):void
- getA():double
- setA(double):void
- isCrashed():boolean
- isFlying():boolean
- land():void
- crash():void
- decFuel(double):void
- getNumberActions():int
- doAction(int):void

7.

3.17.4. Статические открытые данные

Отсутствуют.

3.17.5. Конструктор(ы)

- Plate(Vector position, Vector speed, double initFuel) – задает начальные значения: позицию, скорость, запас топлива.

- position – стартовая позиция;
- speed – начальная скорость;
- initFuel – начальный запас топлива.

3.17.6. Методы

- getPosition():Vector – возвращает позицию модели беспилотного летательного аппарата;
- setPosition(Vector position):void – устанавливает позицию модели беспилотного летательного аппарата;
  - position – новая позиция аппарата.
- getSpeed():Vector – возвращает скорость модели беспилотного летательного аппарата;
- setSpeed(Vector speed):void – устанавливает скорость модели беспилотного летательного аппарата;
  - speed – новая скорость аппарата.
- getFuel():double – возвращает запас топлива модели беспилотного летательного аппарата;
- getQ():double – возвращает текущий расход топлива модели беспилотного летательного аппарата;
- setQ(double q):void – устанавливает текущий расход топлива модели беспилотного летательного аппарата;
  - q – новый расход топлива.
- getA():double – возвращает угол поворота;
- setA(double a):void – устанавливает угол поворота;
  - a – угол поворота.
- isCrashed():boolean – возвращает верно ли, что модель беспилотного летательного аппарата разбилась;
- isFlying():boolean – возвращает верно ли, что модель беспилотного летательного аппарата находится в полете;
- land():void – переводит модель беспилотного летательного аппарата в состояние «Приземлилась»;
- crash():void – переводит модель беспилотного летательного аппарата в состояние «Авария»;
- decFuel(double q):void – уменьшает запас топлива;
  - q – изменение расхода топлива.
- getNumberActions():int – возвращает число возможных выходных воздействий;
- doAction(int index):void – применяет выходное воздействие.
  - index – номер выходного воздействия.

Объявления и описания членов класса находятся в файле:

- \src\plates\Plate.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 1 GB, не менее;
- отображающее устройство (монитор) с поддержкой разрешения 1024x768;
- устройства ввода клавиатура и мышь (трекбол, тачпад);

5. ВЫЗОВ И ЗАГРУЗКА

public Fitness()
public Fitness.fitness(Automaton):double
public Fitness.fitness2(Automaton):double
private Fitness.readOpponent(String):Manager

private State():State
public State(int[], boolean[][], boolean[]) :State
public State.mutate(Random):State
public State.crossover(State, Random):State[]
public State.choosePred(State, State, State, Random):void
public State.setEndState(int, int):State
public State.getEndState(int):int
public State.getActions(int):boolean[]
public State.getVariable(int):int
public State.clone():State

public TableAutomaton(int, int, int, int, int)
public TableAutomaton.getNumberVariables():int
public TableAutomaton.getNumberStates():int
public TableAutomaton.getInitialState():int
public TableAutomaton.mutate(Random):TableAutomaton
public TableAutomaton.crossover(Individual, Random):TableAutomaton[]
public TableAutomaton.doTurn(boolean[]):void
public TableAutomaton.setState(int, State):void
public TableAutomaton.getState(int):State
public TableAutomaton.repairedAutomaton():TableAutomaton
public TableAutomaton.clone():Automaton
public TableAutomaton.toString():String
private TableAutomaton.toIntString(int[]):String
private TableAutomaton.toBoolString(boolean[]):String

public TableAutomatonFactory(int numberStates, int countAllVariables, int numberVariables, int numberActions, double pAction) :TableAutomatonFactory
public TableAutomatonFactory.randomIndividual():TableAutomaton

public TreeAutomaton(int, int, int, TreeAutomatonFactory)
public TreeAutomaton.doTurn(boolean[]):void
public TreeAutomaton.repairedAutomaton():TreeAutomaton
public TreeAutomaton.setState(int, Tree):void
public TreeAutomaton.getState(int):Tree
public TreeAutomaton.getInitialState():int
public TreeAutomaton.getNumberStates():int
public TreeAutomaton.getCountAllVariables():int
public TreeAutomaton.getNumberActions():int
public TreeAutomaton.getHeight():int
public TreeAutomaton.clone():TreeAutomaton
public TreeAutomaton.mutate(Random):TreeAutomaton
public TreeAutomaton.crossover(Individual, Random):TreeAutomaton[]
public TreeAutomaton.fitness():double
public TreeAutomaton.toString():String

public TreeAutomatonFactory(int, int, int, double, int)
public TreeAutomatonFactory.getCountAllVariables():int
public TreeAutomatonFactory.getNumberActions():int
public TreeAutomatonFactory.randomIndividual():TreeAutomaton
public TreeAutomatonFactory.randomTree(int):TreeNode

public AgressiveManager.doTurn():void
public AgressiveManager.getPlates():List
public AgressiveManager.init(List, List):void
public AgressiveManager()

public AutomatonManager.doTurn():void
public AutomatonManager.getPlates():List
public AutomatonManager.init(List, List):void
public AutomatonManager(Automaton)

public Vector(double, double)
public Vector.getLength():double
public Vector.multiply(double):Vector
public Vector.multiply(Vector):double
public Vector.add(Vector):Vector
public Vector.subtract(Vector):Vector
public Vector.rotate(double):Vector

public Competition(Manager, Manager)
public Competittion.getResult(int):double
private Competition.genRandomCoordinates(int):double[]

public Config.getInfluenceDistance():double
public Config.getTimeStep():double
public Config.getConstantT():double
public Config.getConstantF1():double
public Config.getConstantF2():double
public Config.getFieldHeight():int
public Config.getFieldWidth():int
public Config.getPlateDiameter():int
public Config.getPlatesCount():int
public Config.getMaximalRotateAngle():int
public Config.getInitFuel():double
public Config.getInitSpeed():double

public GameLogic(List)
public GameLogic.processSlowPlates():void
public GameLogic.calculateNewSpeeds():void
public GameLogic.movePlates():void
public GameLogic.processFlyingOutPlates():void
public GameLogic.gameOver():boolean

public Plate(Vector, Vector, double)
public Plate.getPosition():Vector
public Plate.setPosition(Vector):void

public Plate.getSpeed():Vector
public Plate.setSpeed(Vector):void
public Plate.getFuel():double
public Plate.getQ():double
public Plate.setQ(double):void
public Plate.getA():double
public Plate.setA(double):void
public Plate.isCrashed():boolean
public Plate.isFlying():boolean
public Plate.land():void
public Plate.crash():void
public Plate.decFuel(double):void
public Plate.getNumberActions():int
public Plate.doAction(int):void

6. ВХОДНЫЕ ДАННЫЕ

- Manager.doTurn():void
  - нет
- Manager.getPlates():List
  - нет
- Manager.init(List<Plate> plates, List<Plate> allPlates):void
  - plates – список беспилотных летательных объектов, являющихся сокомандниками;
  - allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

- Automaton.fitness():double
  - нет
- Automaton.compareTo(Individual o):int
  - o – сравниваемая особь
- Automaton.getPlate():
  - нет
- Automaton.setPlate(Plate plate):void – задает модель беспилотного летательного аппарата, которой будет осуществляться управление.
  - plate – модель беспилотного летательного аппарата, которой будет осуществляться управление

- Fitness.fitness(Automaton a):double
  - a – автомат у которого подсчитывается функция приспособленности
- Fitness.fitness2(Automaton a):double
  - a – автомат у которого подсчитывается функция приспособленности

- State.State(int[] endState, boolean[][] transitionTable, boolean[] variable)
  - endState – массив номеров конечных состояний;
  - transitionTable – матрица переходов;
  - variable – массив значимых переменных.
- State.State()
  - нет

- State.mutate(Random r):State
  - r – генератор случайных чисел.
- State.crossover(State p, Random r):State[]
  - p – второе состояние для скрещивания;
  - r – генератор случайеых чисел.
- State.choosePred(State par, State ch1, State ch2, Random r):void
  - par – один из “родителей”;
  - ch1 – один из “детей”;
  - ch2 – один из “детей”;
  - r – генератор случайных чисел.
- State.setEndState(int i, int en):State

  - i – номер комбинации входных переменных;
  - en – номер конечного состояния.
- State.getEndState(int i):int
  - i – номер комбинации входных переменных.
- State.getActions(int i):boolean[]
  - i – номер комбинации входных переменных.
- State.getVariable(int i):int
  - i – номер опращиваемой переменной.
- State.clone():State
  - нет

- TableAutomaton.TableAutomaton(int numberStates, int numberActions, int numberVariables, int countAllVariables, int is)
  - numberStates – число состояний автомата;
  - numberActions – число возможных вывоздных действий;
  - numberVariables – число знамых переменных;
  - countAllVariables – число возможный входных переменных;
  - is – номер стартового состояния.

- TableAutomaton.getNumberVariables():int
  - нет
- TableAutomaton.getNumberStates():int
  - нет
- TableAutomaton.getInitialState():int
  - нет
- TableAutomaton.mutate(Random r):TableAutomaton
  - r – генератор случайных чисел;
- TableAutomaton.crossover(Individual p, Random r):TableAutomaton[]
  - p – особь для скрещивания;
  - r – генератор случайных чисел;
- TableAutomaton.doTurn(boolean[] variables):void
  - variables – массив значений входных переменных;
- TableAutomaton.setState(int i, State s):void
  - i – номер заменяемого состояния;
  - s – новое состояние;
- TableAutomaton.getState(int i):State – возвращает состояние по номеру;
  - i – номер состояния;
- TableAutomaton.repairedAutomaton():TableAutomaton
  - нет
- TableAutomaton.clone():Automaton
  - нет
- TableAutomaton.toString():String
  - нет

- TableAutomatonFactory.TableAutomatonFactory(int numberStates, int countAllVariables, int numberVariables, int numberActions, double pAction)
  - numberStates – число состояний автомата;
  - numberActions – число возможных вывоздных действий;

  - numberVariables – число знамых переменных;
  - countAllVariables – число возможный входных переменных;
  - pAction – вероятность действия на переходе.

- TableAutomatonFactory.randomIndividual():TableAutomaton
  - нет

- TableAutomatonFactoryLoader.TableAutomatonFactoryLoader(JarFile jr)
  - jr – jar-архив, в котором находится фабрика;

- TableAutomatonFactoryLoader.load():TableAutomatonFactory
  - нет
- TableAutomatonFactoryLoader.getProperties():Properties
  - нет

- TreeAutomatonFactoryLoader.TreeAutomatonFactoryLoader(JarFile jr)
  - jr – jar-архив, в котором находится фабрика;

- TreeAutomatonFactoryLoader.load():TreeAutomatonFactory
  - нет
- TreeAutomatonFactoryLoader.getProperties():Properties
  - нет

- TreeAutomaton.TreeAutomaton(int numberStates, int numberVariables, int number Actions, TreeAutomatonFactory factory)
  - numberStates – число состояний автомата;
  - numberActions – число возможных вывоздных действий;
  - numberVariables – число знамых переменных;
  - factory – генератор случайных деревьев решений;

- TreeAutomaton.doTurn(boolean[] variables):void
  - variables – значения входных переменных;
- TreeAutomaton.setState(int i, Tree t):void
  - i – номер заменяемого состояния;
  - t – новое состояние(дерево решений);
- TreeAutomaton.getState(int i):Tree
  - i – номер состояния;
- TreeAutomaton.getInitialState():int
  - нет
- TreeAutomaton.getNumberStates():int
  - нет
- TreeAutomaton.getCountAllVariables():int
  - нет
- TreeAutomaton.getNumberActions():int
  - нет
- TreeAutomaton.getHeight():int

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

- TreeAutomatonFactory.TreeAutomatonFactory(int numberStates, int countAllVariables, int numberActions, double pAct, int height) – задает число состояний, число переменных, число выходных воздействий, вероятность появления каждого выходного воздействия и рекомендуемую высоту дерева решений генерируемых автоматов.
  - numberStates – число состояний автомата;
  - numberActions – число возможных вывоздных действий;
  - numberVariables – число знамых переменных;
  - pAct – вероятность действия на переходе;
  - height - рекоммендуемая высота дерева решений.

- TreeAutomatonFactory.getCountAllVariables():int
  - нет
- TreeAutomatonFactory.getNumberActions():int
  - нет
- TreeAutomatonFactory.randomIndividual():TreeAutomaton
  - нет
- TreeAutomatonFactory.randomTree(int h):TreeNode
  - h - высота уже сгенерированной части

- AgressiveManager.doTurn():void
- AgressiveManager.getPlates():List
- AgressiveManager.init(List<Plate> plates, List<Plate> allPlates):void
  - plates – список беспилотных летательных объектов, являющихся сокомандниками;
  - allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

- AutomatonManager.AutomatonManager(Automaton a) – задает автоматы для всех моделей беспилотных летательных аппаратов.
  - a – управляющий автомат.

- AutomatonManager.doTurn():void
  - нет
- AutomatonManager.getPlates():List

- нет
- AutomatonManager.init(List<Plate> plates, List<Plate> allPlates):void
  - plates – список беспилотных летательных объектов, являющихся сокомандниками;
  - allPlates – список всех беспилотных летательных объектов, участвующих в соревновании;

- Vector.Vector(double x, double y)
  - x – абсцисса;
  - y – ордината.

- Vector.getLength():double
  - нет
- multiply(double k):Vector
  - k – множитель.
- Vector.multiply(Vector v):double
  - v – множитель.
- Vector.add(Vector v):Vector
  - v – слагаемое.
- Vector.subtract(Vector v):Vector
  - v - вычитаемое
- Vector.rotate(double phi):Vector
  - phi – угол поворота

- Competition.Competition(Manager man1, Manager man2) – задает системы управления моделями беспилотных летательных аппаратов для двух команд;
  - man1 – тестируемая команда.
  - man2 – соперник.
- Competition.genRandomCoordinates(int index):double[]
  - index – номер модели.
- Competition.getResult(int index):double
  - index – номер команды.
- emulate():void
  - нет

- Config.getInfluenceDistance():double
  - нет
- Config.getTimeStep():double
  - нет
- Config.getConstantT():double
  - нет
- Config.getConstantF1():double
  - нет
- Config.getConstantF2():double
  - нет
- Config.getFieldHeight():int
  - нет

- Config.getFieldWidth():int
  - нет
- Config.getPlateDiameter():int
  - нет
- Config.getPlatesCount():int
  - нет
- Config.getMaximalRotateAngle():int
  - нет
- Config.getInitFuel():double
  - нет
- Config.getInitSpeed():double
  - нет

- GameLogic.GameLogic(List<Plates> plates)
  - plates – участвующие модели.

- GameLogic.processSlowPlates():void
  - нет
- GameLogic.calculateNewSpeeds():void
  - нет
- GameLogic.movePlates():void
  - нет
- GameLogic.processFlyingOutPlates():void
  - нет
- GameLogic.gameOver():boolean
  - нет

- Plate.Plate(Vector position, Vector speed, double initFuel) – задает начальные значения: позицию, скорость, запас топлива.
  - position – стартовая позиция;
  - speed – начальная скорость;
  - initFuel – начальный запас топлива.

- Plate.getPosition():Vector – возвращает позицию модели беспилотного летательного аппарата;
  - нет
- Plate.setPosition(Vector position):void
  - position – новая позиция аппарата.
- Plate.getSpeed():Vector
  - нет
- Plate.setSpeed(Vector speed):void
  - speed – новая скорость аппарата.
- Plate.getFuel():double
  - нет
- Plate.getQ():double
  - нет
- Plate.setQ(double q):void
  - q – новый расход топлива.

- Plate.getA():double
  - нет
- Plate.setA(double a):void
  - a – угол поворота.
- Plate.isCrashed():boolean
  - нет
- Plate.isFlying():boolean
  - нет
- Plate.land():void
  - нет
- Plate.crash():void
  - нет
- Plate.decFuel(double q):void
  - q – изменение расхода топлива.
- Plate.getNumberActions():int
  - нет
- Plate.doAction(int index):void
  - index – номер выходного воздействия.

7. ВЫХОДНЫЕ ДАННЫЕ

- Manager.doTurn():void – нет;
- Manager.getPlates():List – возвращает список моделей беспилотных летательных аппаратов;
- Manager.init(List<Plate> plates, List<Plate> allPlates):void – нет.

- Automaton.fitness():double – возвращает значение функции приспособленности;
- Automaton.compareTo(Individual o):int – возвращает результат сравнения функции приспособленности со значением функции приспособленности другой особи;
- Automaton.getPlate():Plate – возвращает модель беспилотного летательного аппарата, которой осуществляется управление;
- Automaton.setPlate(Plate plate):void – нет.

- Fitness.fitness(Automaton a):double – возвращает результат подсчета значения функции приспособленности в десяти соревнованиях;
- Fitness.fitness2(Automaton a):double – возвращает результат подсчета значения функции приспособленности в тридцати соревнованиях;

- State.mutate(Random r):State – возвращает состояние после применения к нему операции мутации;
- State.crossover(State p, Random r):State[] – возвращает два состояния, получаемые в результате скрещивания данного и передаваемого методу в качестве параметра;
- State.choosePred(State par, State ch1, State ch2, Random r):void – нет;
- State.setEndState(int i, int en):State – возвращает состояние с установленным номером состояния, в которое ведет переход из данного состояния при заданной комбинации значений значимых переменных;
- State.getEndState(int i):int – возвращает номер состояния, в которое ведет переход из данного состояния, по комбинации значений значимых переменных;
- State.getActions(int i):boolean[] – возвращает список действий, которые выполняются при выборе перехода из данного состояния, по комбинации значений значимых переменных;
- State.getVariable(int i):int – возвращает является ли данная переменная значимой;
- State.clone():State – возвращает копию данного состояния.

- TableAutomaton.getNumberVariables():int – возвращает число переменных;
- TableAutomaton.getNumberStates():int – возвращает число состояний;
- TableAutomaton.getInitialState():int – возвращает номер начального состояния;
- TableAutomaton.mutate(Random r):TableAutomaton – возвращает автомат, к которому применена операция мутации;
- TableAutomaton.crossover(Individual p, Random r):TableAutomaton[] – возвращает два автомата, получаемых в результате применения операции скрещивания к данному и переданному автоматам в качестве параметра;
- TableAutomaton.doTurn(boolean[] variables):void – нет;
- TableAutomaton.setState(int i, State s):void – нет;

- TableAutomaton.getState(int i):State – возвращает состояние по номеру;
- TableAutomaton.repairedAutomaton():TableAutomaton – применяет к автомату алгоритм восстановления связей;
- TableAutomaton.clone():Automaton – возвращает копию автомата;
- TableAutomaton.toString():String – возвращает строковое представление автомата.

- TableAutomaton.randomIndividual():TableAutomaton – возвращает случайно сгенерированный автомат в представлении с помощью сокращенных таблиц переходов.

- TableAutomatonFactoryLoader.load():TableAutomatonFactory – возвращает новую фабрику автоматов;
- TableAutomatonFactoryLoader.getProperties():Properties – возвращает список настраиваемых параметров алгоритма;

- TreeAutomatonFactoryLoader.load():TreeAutomatonFactory – возвращает новую фабрику автоматов;
- TreeAutomatonFactoryLoader.getProperties():Properties – возвращает список настраиваемых параметров алгоритма;

- TreeAutomaton.doTurn(boolean[] variables):void – нет;
- TreeAutomaton.setState(int i, Tree t):void – нет;
- TreeAutomaton.getState(int i):Tree – возвращает состояние по номеру;
- TreeAutomaton.getInitialState():int – возвращает номер начального состояния;
- TreeAutomaton.getNumberStates():int – возвращает число состояний;
- TreeAutomaton.getCountAllVariables():int – возвращает число переменных;
- TreeAutomaton.getNumberActions():int – возвращает число действий;
- TreeAutomaton.getHeight():int – возвращает рекомендуемую высоту дерева решений;
- TreeAutomaton.clone():TreeAutomaton – возвращает копию автомата;
- TreeAutomaton.mutate(Random r):TreeAutomaton – возвращает автомат, к которому применена операция мутации;
- TreeAutomaton.crossover(Individual p, Random r):TreeAutomaton[] – возвращает результат применения операции скрещивания к данному автомату и автомату, переданному в качестве параметра;
- TreeAutomaton.fitness():double – возвращает значение функции приспособленности;
- TreeAutomaton.toString():String – возвращает строковое представление автомата.

- TreeAutomatonFactory.getCountAllVariables():int – возвращает число переменных;
- TreeAutomatonFactory.getNumberActions():int – возвращает число выходных воздействий;
- TreeAutomatonFactory.randomIndividual():TreeAutomaton – возвращает случайно сгенерированный автомат, представленный с помощью деревьев решений;

- TreeAutomatonFactory.randomTree(int h):TreeNode – возвращает случайно сгенерированное дерево решений;

- AgressiveManager.doTurn():void – вырабатывает очередные выходные воздействия на управляемые модели беспилотных летательных аппаратов;
- AgressiveManager.getPlates():List – возвращает список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
- AgressiveManager.init(List<Plate> plates, List<Plate> allPlates):void – нет.

- AutomatonManager.doTurn():void – вырабатывает очередные выходные воздействия на управляемые модели беспилотных летательных аппаратов;
- AutomatonManager.getPlates():List – возвращает список моделей беспилотных летательных аппаратов, которыми осуществляется управление;
- AutomatonManager.init(List<Plate> plates, List<Plate> allPlates):void – нет.

- Vector.getLength():double – возвращает длину вектора;
- Vector.multiply(double k):Vector – возвращает результат умножения вектора на число;
- Vector.multiply(Vector v):double – возвращает скалярное произведение с вектором, переданным в качестве параметра;
- Vector.add(Vector v):Vector – возвращает результат сложения векторов;
- Vector.subtract(Vector v):Vector – возвращает результат вычитания векторов;
- Vector.rotate(double phi):Vector – возвращает результат поворота вектора.

- Competition.genRandomCoordinates(int index):double[] – случайным образом расставляет модели беспилотных летательных аппаратов;
- Competition.getResult(int index):double – возвращает результат соревнования;
- Competition.emulate():void – нет.

- Config.getInfluenceDistance():double – возвращает максимальное расстояние, на котором проявляется аэродинамическое взаимодействие;
- Config.getTimeStep():double – возвращает шаг по времени;
- Config.getConstantT():double – возвращает константу *T*;
- Config.getConstantF1():double – возвращает константу *F1*;
- Config.getConstantF2():double – возвращает константу *F2*;
- Config.getFieldHeight():int – возвращает высоту поля;
- Config.getFieldWidth():int – возвращает ширину поля;
- Config.getPlateDiameter():int – возвращает диаметр модели беспилотного летательного аппарата;
- Config.getPlatesCount():int – возвращает число моделей беспилотных летательных аппаратов в каждой команде;
- Config.getMaximalRotateAngle():int – возвращает максимальный угол поворота;
- Config.getInitFuel():double – возвращает начальное количество топлива;

- Config.getInitSpeed():double – возвращает начальную скорость

- GameLogic.processSlowPlates():void – нет;
- GameLogic.calculateNewSpeeds():void – нет;
- GameLogic.movePlates():void – нет;
- GameLogic.processFlyingOutPlates():void – нет;
- GameLogic.gameOver():boolean – возвращает закончилось ли соревнование.

- Plate.getPosition():Vector – возвращает позицию модели беспилотного летательного аппарата;
- Plate.setPosition(Vector position):void – нет;
- Plate.getSpeed():Vector – возвращает скорость модели беспилотного летательного аппарата;
- Plate.setSpeed(Vector speed):void – нет;
- Plate.getFuel():double – возвращает запас топлива модели беспилотного летательного аппарата;
- Plate.getQ():double – возвращает текущий расход топлива модели беспилотного летательного аппарата;
- Plate.setQ(double q):void – нет;
- Plate.getA():double – возвращает угол поворота;
- Plate.setA(double a):void – нет;
- Plate.isCrashed():boolean – возвращает верно ли, что модель беспилотного летательного аппарата разбилась;
- Plate.isFlying():boolean – возвращает верно ли, что модель беспилотного летательного аппарата находится в полете;
- Plate.land():void – нет;
- Plate.crash():void – нет;
- Plate.decFuel(double q):void – нет;
- Plate.getNumberActions():int – возвращает число возможных выходных воздействий;
- Plate.doAction(int index):void – нет.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм.	*Измененных*	*Замененных*	*Новых*	*Аннулированных*					