Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра: электронных вычислительных машин

Факультет: компьютерных систем и сетей

**ОТЧЕТ**

к лабораторной работе на тему:

«Змейка»

Поверил: Выполнил:

Кухарчук И. В. ст. гр. 350505

Садовский А. В.

МИНСК 2015

Скриншоты:

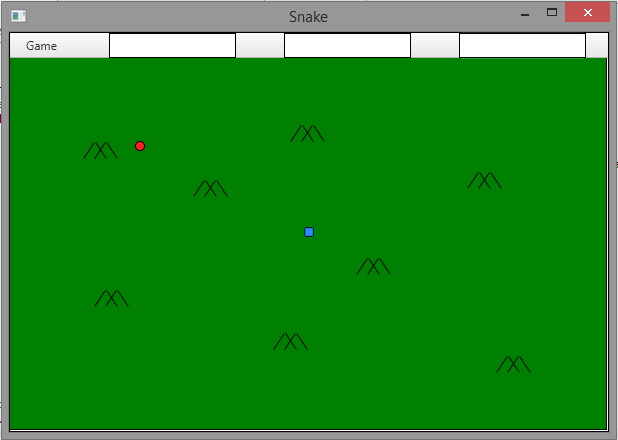


Рисунок 1 – Главное окно

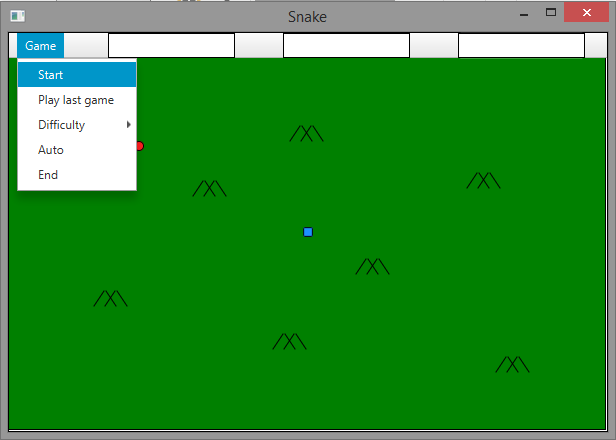
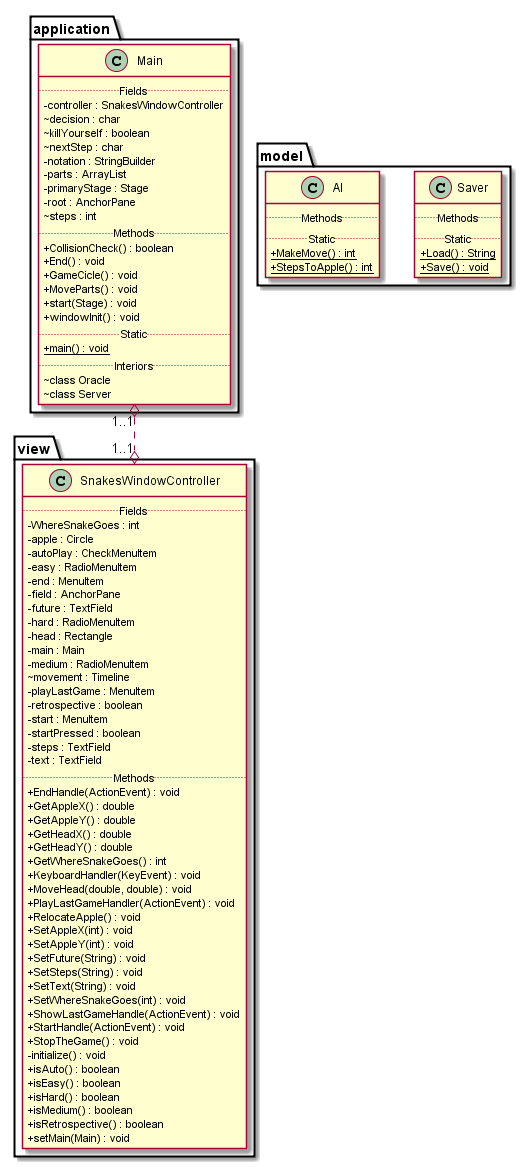


Рисунок 2 – Главное окно с открытым меню

Блок-схема метода GameCicle:



Диаграмма классов:



Обоснование компоновки классов:

Мною было выделено три основных пакета application, model и view. Application отвечает за управление процессом игры и содержит основные принципы и механизмы, жизненно важные для игры, model – за сложные автоматические вспомогательные механизмы вызываемые при необходимости, в данном случае «искусственный интеллект» и сохранение в файл, а view – за отображение данных на экран, а также управление ими.

Была выбрана именно данная компоновка классов в основном из-за ориентации на удобство и скорость создания приложения, а также, частично, быстродействие конечного продукта. Создав жесткую зависимость между классами Main и SnakeWindowController, я добился удобного доступа к ресурсам обоих классов и предупредил появление дополнительных методов обмена данными между экземплярами данных классов. Однако, данный подход не рекомендуется использовать в больших проектах, так как жесткая зависимость между классами приводит к тому, что изменение одного влечет изменение другого, что приводит к появлению дополнительных проблем в ходе разработки.

С классами пакета model другие классы жестко не связаны и только используют их методы при необходимости, в частности при сохранении нотации в файл. Классы этого пакета основаны на несложных алгоритмах и несут в себе вспомогательные функции, которые были бы излишне громоздкими для помещения их в другие классы. Также механизмы, описанные в них, должны иметь возможность использоваться повсеместно и их обособление как нельзя лучше этому способствует.

С введением в проект многопоточности в класс main добавились новые внутренние классы. Это классы Oracle и Server. Эти классы расширяют класс Thread и представляют собой отдельные потоки. Наиболее важным является класс Server: именно он в последствии создает поток Oracle. Потоки этих классов имеют ключевое значение в приложении, но могут использоваться только в контексте класса main, т. к. для взаимодействия данных потоков с потоком клиентом, описанного далее, им необходимы переменные.

Нотация и её обоснование:

Разработанная мной нотация содержит следующий набор условных обозначений:

Ax,y – соответствует перемещению яблока в координаты x и y;

U – перемещение головы змеи вверх;

R – перемещение головы змеи вправо;

D – перемещение головы змеи вниз;

L – перемещение головы змеи влево;

E – конец игры.

Данная нотация представляет собой строку символов без пробелов, описывающую последовательность действий в игре.

Мной был выбран именно данный вид нотации в связи с его простотой и удобством. Данные в строку легко помещать и также легко из неё их изымать. Отсутствие пробелов упрощает забор данных и значительно уменьшает объем строки. Таким образом, последовательно изымая символы можно восстановить последовательность событий прошлой игры с абсолютной точностью и без заметных задержек времени на поиск данных, так как только перевод координат нового положения «яблока» требует обработки, тогда как все остальные данные сами в себе несут тот контекст в котором их необходимо понимать. Трудность составляет только обработка таких данных человеком непосредственно через текстовый редактор, такой как блокнот, в связи с тем, что из-за отсутствия разделительных переходов на новую строку текст может сливаться и быть трудно читаем в больших объемах, однако каждое событие в игре происходит с определённым временным интервалом (по умолчанию 125 миллисекунд) и зная время, можно с лёгкостью высчитать номер того шага, который нам будет интересен.

Клиент-серверное взаимодействие

Клиент-серверное взаимодействие в данном проекте реализовано таким образом, что потоком-клиентом является основной поток JavaFX приложения. Поток-сервер, в свою очередь, вызывается на старте программы в качестве потока-демона и просчитывает следующее состояние системы отображения на основе текущего состояния, а также принятых игроком решений. Изменения вступают в силу при каждом запросе со стороны потока-клиента. Такой тип взаимодействия клиентского и серверного процессов позволил эффективно производить вычисления и теперь, при низких уровнях задержки анимации вычисления не останавливают сам процесс, а дополняют его с большей эффективностью. Также важно отметить взаимодействие процесса-клиента и процесса-сервера. Несмотря на то, что расчеты процесса-сервера зависят от текущего состояния клиента, они проводятся с большей частотой, ограниченной лишь производительностью системы. Данные, полученные из расчетов, сохраняются в специальной переменной nextStep, изменение которой отслеживается клиентом и применяется к интерфейсу.

Также из потока-сервера отдельно вызывается поток-«предсказатель». «Предсказатель» использует уже имеющуюся имитацию искусственного интеллекта для предсказания следующего хода игрока, а также подсчитывает количество шагов до цели. Сразу после подсчета, полученные данные помещаются в соответствующие переменные, для информирования клиента. Таким образом, исходя из предположения, что игроком будет предпринят наиболее простой и эффективный шаг (а именно на таком соображении организуется имитация искусственного интеллекта в игре), в качестве предполагаемого значения выступает ход, который был бы принят компьютером, если бы он управлял ходом игры. Расчет же количества шагов до цели является тривиальной задачей деления разности координат змеи и яблока на ширину шага. Производительности «предсказателя» не ограничена програмно и зависит только от производительности системы на которой запущена. Таким образом, на процессорах с большей частотой «предсказатель» будет работать эффективнее. В случае процессора с частотой 2.3 ГГц, работающего на 100%, производительность будет равна частоте за вычетом ресурсов необходимых JVM и вызовам функций. Что приблизительно равно 2300000000 предсказаний в секунду.



Сортировка сохранённых игр

Для реализации сортировки сохранённых игр мною был выбран единственный критерий, который и только он говорит об успешности проведённых игр: количество съеденных яблок. Ни время проведённое в игре, ни количество сделанных шагов не могут говорить о том, какая игра была лучше, а какая хуже. Игра также примечательна тем, что нет финальной цели, достигнув которой, можно было бы судить о выигрыше. Уровни также не изменяются и критерием это служить не может. Но вот количество съеденных яблок, и соответственно длина змеи, как нельзя полно отражает то, насколько удачной была игра. Однако мной также была реализована и сортировка по длительности игры, и по совместительству количеству шагов. Данный критерий не может считаться официальным показателем успеха в игре, но, тем не менее, такой критерий имеет место быть и отлично дополняет основную сортировку.

Сравнивая эффективность сортировок на языках Scala и Java я пришел к выводу, что эффективность данных языков наиболее полно раскрывается только при определенных, благоприятных для данного языка, а также выбранного для него алгоритма, обстоятельствах. В случае моего проекта, выявилась неоднозначность доминирования одного из языков. Абсолютного лидера выявить не удалось, однако мною были получены данные, приближаясь к которым языки меняют свои позиции в первенстве. При объемах нотации до 500 kB с небольшим опережением, порядка 2-3 миллисекунд, идет Java, но по превышении данной границы ситуация обращается и лидерство завоевывает Scala. Примечательно, что отрыв в производительности растет по мере увеличения объема данных, но не значительно. При объеме данных свыше 2 MB разница составила 20 миллисекунд. Стоит отметить также, что с ростом объёмов данных росли и искажения в замеряемом времени. На объемах до 100 kB величина отклонялась на 1-2 значения, при объемах свыше 2 MB – 10-15.

**ПРИЛОЖЕНИЕ**

Самогенерируемая документация:

application

## Class Main

* java.lang.Object
  + javafx.application.Application
    - application.Main

public class **Main**

extends javafx.application.Application

Основной класс приложения Организует работу

**Version:**

1.0

**Author:**

Sanyadovskiy

### *Nested Class Summary*

### Nested classes/interfaces inherited from class javafx.application.Application

javafx.application.Application.Parameters

### *Field Summary*

### Fields inherited from class javafx.application.Application

STYLESHEET\_CASPIAN, STYLESHEET\_MODENA

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Main**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#Main--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| boolean | [**CollisionCheck**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#CollisionCheck--)()  проверка на столкновение с краем и хвостом |
| void | [**End**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#End--)()  конец игры |
| void | [**GameCicle**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#GameCicle--)()  Основной цикл игры На основе данных, полученных от потока-сервера, осуществляет изменение состояния системы. |
| static void | [**main**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#main-java.lang.String:A-)(java.lang.String[] args) |
| void | [**MoveParts**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#MoveParts--)()  сдвиг хвоста |
| void | [**start**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#start-javafx.stage.Stage-)(javafx.stage.Stage primaryStage) |
| void | [**windowInit**](file:///F:\eclipse\work\KPP1\doc\application\Main.html#windowInit--)()  инициализация окна |

### Methods inherited from class javafx.application.Application

getHostServices, getParameters, getUserAgentStylesheet, init, launch, launch, notifyPreloader, setUserAgentStylesheet, stop

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Main

public Main()

### *Method Detail*

#### start

public void start(javafx.stage.Stage primaryStage)

**Specified by:**

start in class javafx.application.Application

#### windowInit

public void windowInit()

инициализация окна

#### main

public static void main(java.lang.String[] args)

#### GameCicle

public void GameCicle()

Основной цикл игры На основе данных, полученных от потока-сервера, осуществляет изменение состояния системы. Вызывает ИИ. Воспроизводит последнюю игру

#### End

public void End()

конец игры

#### CollisionCheck

public boolean CollisionCheck()

проверка на столкновение с краем и хвостом

**Returns:**

было ли столкновение

#### MoveParts

public void MoveParts()

сдвиг хвоста

model

## Class AI

* java.lang.Object
  + model.AI

public class **AI**

extends java.lang.Object

Класс отвечающий за "искуственный интеллект" Определяет поведение игрока-компьютера

**Version:**

1.0

**Author:**

Sanyadovskiy

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**AI**](file:///F:\eclipse\work\KPP1\doc\model\AI.html#AI--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| static int | [**MakeMove**](file:///F:\eclipse\work\KPP1\doc\model\AI.html#MakeMove-double-double-double-double-int-)(double appleX, double appleY, double headX, double headY, int direction)  Принятие решения о следующем шаге |
| static int | [**StepsToApple**](file:///F:\eclipse\work\KPP1\doc\model\AI.html#StepsToApple-double-double-double-double-)(double appleX, double appleY, double headX, double headY)  Подсчитывает количество шагов до цели |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### AI

public AI()

### *Method Detail*

#### MakeMove

* + - public static int MakeMove(double appleX,
    - double appleY,
    - double headX,
    - double headY,

int direction)

Принятие решения о следующем шаге

**Parameters:**

appleX - X координата яблока

appleY - Y координата яблока

headX - X координата головы змеи

headY - Y координата головы змеи

direction - направление движения змеи

**Returns:**

следующий шаг

#### StepsToApple

* + - public static int StepsToApple(double appleX,
    - double appleY,
    - double headX,

double headY)

Подсчитывает количество шагов до цели

**Parameters:**

appleX - X координата яблока

appleY - Y координата яблока

headX - X координата головы змеи

headY - Y координата головы змеи

**Returns:**

количество шагов до цели

## Class Saver

* java.lang.Object
  + model.Saver

public class **Saver**

extends java.lang.Object

Класс для сохранения сток в файл и их чтения

**Version:**

1.0

**Author:**

Sanyadovskiy

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Saver**](file:///F:\eclipse\work\KPP1\doc\model\Saver.html#Saver--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| static java.lang.String | [**Load**](file:///F:\eclipse\work\KPP1\doc\model\Saver.html#Load-java.lang.String-)(java.lang.String from)  Загрузка строки из файла |
| static void | [**Save**](file:///F:\eclipse\work\KPP1\doc\model\Saver.html#Save-java.lang.String-java.lang.String-)(java.lang.String what, java.lang.String where)  Сохранение в файл |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Saver

public Saver()

### *Method Detail*

#### Save

* + - public static void Save(java.lang.String what,

java.lang.String where)

Сохранение в файл

**Parameters:**

what - Строка, которую сохраняем

where - Строка, содержащая путь к файлу

#### Load

public static java.lang.String Load(java.lang.String from)

Загрузка строки из файла

**Parameters:**

from - Строка, содержащая путь к файлу

**Returns:**

Строку, полученную из файла, либо путь к файлу в случае ошибки

view

## Class SnakesWindowController

* java.lang.Object
  + view.SnakesWindowController

public class **SnakesWindowController**

extends java.lang.Object

Класс контроллер Помещает в себе данные, загруженные из fxml файла

**Version:**

1.0

**Author:**

Sanyadovskiy

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**SnakesWindowController**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SnakesWindowController--)()  конструктор по умолчанию |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**EndHandle**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#EndHandle-javafx.event.ActionEvent-)(javafx.event.ActionEvent e) |
| double | [**GetAppleX**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#GetAppleX--)() |
| double | [**GetAppleY**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#GetAppleY--)() |
| double | [**GetHeadX**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#GetHeadX--)() |
| double | [**GetHeadY**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#GetHeadY--)() |
| int | [**GetWhereSnakeGoes**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#GetWhereSnakeGoes--)() |
| boolean | [**isAuto**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#isAuto--)()  находится ли игра в автоматическом режиме |
| boolean | [**isEasy**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#isEasy--)() |
| boolean | [**isHard**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#isHard--)() |
| boolean | [**isMedium**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#isMedium--)() |
| boolean | [**isRetrospective**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#isRetrospective--)()  Включено ли воспроизведение прошлой игры |
| void | [**KeyboardHandler**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#KeyboardHandler-javafx.scene.input.KeyEvent-)(javafx.scene.input.KeyEvent e) |
| void | [**MoveHead**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#MoveHead-double-double-)(double offsetX, double offsetY)  смещение головы на заданное отклонение |
| void | [**PlayLastGameHandler**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#PlayLastGameHandler-javafx.event.ActionEvent-)(javafx.event.ActionEvent e)  Устанавливает режим вопроизведения последней завершенной игры |
| void | [**RelocateApple**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#RelocateApple--)()  случайное перемещение яблока |
| void | [**SetAppleX**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetAppleX-int-)(int x) |
| void | [**SetAppleY**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetAppleY-int-)(int y) |
| void | [**SetFuture**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetFuture-java.lang.String-)(java.lang.String mess) |
| void | [**setMain**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#setMain-application.Main-)([**Main**](file:///F:\eclipse\work\KPP1\doc\application\Main.html) main)  получение ссылки на main |
| void | [**SetSteps**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetSteps-java.lang.String-)(java.lang.String mess) |
| void | [**SetText**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetText-java.lang.String-)(java.lang.String mess) |
| void | [**SetWhereSnakeGoes**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#SetWhereSnakeGoes-int-)(int arg)  задание направлния движения змеи (головы) |
| void | [**ShowLastGameHandle**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#ShowLastGameHandle-javafx.event.ActionEvent-)(javafx.event.ActionEvent e) |
| void | [**StartHandle**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#StartHandle-javafx.event.ActionEvent-)(javafx.event.ActionEvent e) |
| void | [**StopTheGame**](file:///F:\eclipse\work\KPP1\doc\view\SnakesWindowController.html#StopTheGame--)()  Останавливает таймер |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### SnakesWindowController

public SnakesWindowController()

конструктор по умолчанию

### *Method Detail*

#### setMain

public void setMain([Main](file:///F:\eclipse\work\KPP1\doc\application\Main.html) main)

получение ссылки на main

**Parameters:**

main - ссылка на main класс

#### StartHandle

public void StartHandle(javafx.event.ActionEvent e)

#### PlayLastGameHandler

public void PlayLastGameHandler(javafx.event.ActionEvent e)

Устанавливает режим вопроизведения последней завершенной игры

**Parameters:**

e - событие нажатой кнопки меню

#### ShowLastGameHandle

public void ShowLastGameHandle(javafx.event.ActionEvent e)

#### EndHandle

public void EndHandle(javafx.event.ActionEvent e)

#### StopTheGame

public void StopTheGame()

Останавливает таймер

#### GetAppleX

public double GetAppleX()

#### GetAppleY

public double GetAppleY()

#### SetAppleX

public void SetAppleX(int x)

#### SetAppleY

public void SetAppleY(int y)

#### GetHeadX

public double GetHeadX()

#### GetHeadY

public double GetHeadY()

#### SetText

public void SetText(java.lang.String mess)

#### SetSteps

public void SetSteps(java.lang.String mess)

#### SetFuture

public void SetFuture(java.lang.String mess)

#### MoveHead

* + - public void MoveHead(double offsetX,

double offsetY)

смещение головы на заданное отклонение

**Parameters:**

offsetX - смещение по оси X

offsetY - смешение по оси Y

#### GetWhereSnakeGoes

public int GetWhereSnakeGoes()

#### SetWhereSnakeGoes

public void SetWhereSnakeGoes(int arg)

задание направлния движения змеи (головы)

**Parameters:**

arg - новое направление движения змеи

#### RelocateApple

public void RelocateApple()

случайное перемещение яблока

#### isAuto

public boolean isAuto()

находится ли игра в автоматическом режиме

**Returns:**

значение состояния

#### isRetrospective

public boolean isRetrospective()

Включено ли воспроизведение прошлой игры

**Returns:**

значение состояния

#### isEasy

public boolean isEasy()

#### isMedium

public boolean isMedium()

#### isHard

public boolean isHard()

#### KeyboardHandler

public void KeyboardHandler(javafx.scene.input.KeyEvent e)