Raynor Kuang

Rose

APCS, Period 4

1 June 2011

APCS End-of-Year Assignment Stubs

IMPORTANT--> See Snake class and OptionPanel class fields for data structures

All work was and will be done by Raynor Kuang

**Class SnakeDriver**

java.lang.Object

**SnakeDriver**

public class **SnakeDriver** extends java.lang.Object

The Driver of Snake, where it all begins.

|  |
| --- |
| **Field Detail** |

**frame**

private static javax.swing.JFrame **frame**

Frame of the game.

|  |
| --- |
| **Constructor Detail** |

**SnakeDriver**

public **SnakeDriver**()

|  |
| --- |
| **Method Detail** |

**main**

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

Runs the game.

**initialize**

private static void **initialize**()

Actually initializes the program.

**reload**

public static void **reload**(javax.swing.JPanel j,

java.lang.String s)

Reloads the frame with a new panel.

**Parameters:**

j - JPanel run in the game frame

s - Title of the panel

**Class Snake**

java.lang.Object

**Snake**

public class **Snake**extends java.lang.Object

A Snake class. SEE segments FIELD FOR DATA STRUCTURE

|  |
| --- |
| **Field Detail** |

**segments**

public java.util.ArrayList<Segment> **segments**

A DATA STRUCTURE. The segments that make up the snake. Segments can be added, removed, and gotten from index using this data structure

**mySpeed**

public double **mySpeed**

Speed factor of snake

**myDir**

public double **myDir**

Direction of snake

**myColor**

public java.awt.Color **myColor**

Color of snake

**myLength**

public int **myLength**

Length of snake i.e. number of segments

|  |
| --- |
| **Constructor Detail** |

**Snake**

public **Snake**(java.awt.Color c)

Creates a snake of color c

**Parameters:**

c - The color of the snake

|  |
| --- |
| **Method Detail** |

**move**

public void **move**()

Moves the snake in its current direction

**addSegment**

public void **addSegment**(Segment seg)

Add a segment to the snake and increment length

**Parameters:**

seg - The segment to be added

**removeSegment**

public Segment **removeSegment**()

Remove a segment and decrement length

**Returns:**

The removed segment

**getDir**

public double **getDir**()

Returns the direciton of the snake

**Returns:**

The current direction

**getLength**

public int **getLength**()

Returns the length of the snake

**Returns:**

The current length

**getColor**

public java.awt.Color **getColor**()

Returns the color of the snake

**Returns:**

The current color

**setColor**

public void **setColor**(java.awt.Color c)

Sets the snake's color

**Parameters:**

c - The new color of the snake

**getSegments**

public java.util.ArrayList<Segment> **getSegments**()

Returns the Segments of the snake

**Returns:**

The segments field making up the snake

**getSegment**

public Segment **getSegment**(int k)

Returns the Segment of number k in the segments field

**Parameters:**

k - number of the segment

**Returns:**

A segment of number k

**getHead**

public Segment **getHead**()

Returns first Segment in the snake; identical to calling getSegment(0)

**Returns:**

The first Segment in the segments field

**See Also:**

getSegments(int k)

**turnRight**

public void **turnRight**()

Turns snake right

**turnLeft**

public void **turnLeft**()

Turns snake left

**setDir**

public void **setDir**(double dir)

Changes the snake's direction

**Parameters:**

dir - The new direction of the snake

**getDistance**

public double **getDistance**(int x,

int y)

Returns snake's distance from a point

**Parameters:**

x - x-coordinate of point

y - y-coordinate of point

**Returns:**

distance of snake from point

**Class Food**

java.lang.Object

**Food**

public class **Food**extends java.lang.Object

The Food class; eaten by the Snake

|  |
| --- |
| **Field Detail** |

**myX**

public int **myX**

Coordinates of food

**myY**

public int **myY**

Coordinates of food

**myWorth**

public int **myWorth**

Worth of food

**myCirc**

public java.awt.geom.Ellipse2D.Double **myCirc**

Circle bounded by square with upper left corner myX, myY

|  |
| --- |
| **Constructor Detail** |

**Food**

public **Food**(int x,

int y,

int worth)

Constructor for the food

**Parameters:**

x - x-coordinate of food

y - y-coordinate of food

worth - worth of food

|  |
| --- |
| **Method Detail** |

**getX**

public int **getX**()

Returns x-coordinate of food

**Returns:**

the x coordinate of the food

**getY**

public int **getY**()

Returns y-coordinate of food

**Returns:**

the y coordinate of the food

**getCircle**

public java.awt.geom.Ellipse2D **getCircle**()

Returns the circle defining the food

**Returns:**

the circle bounded by the square with the upper left corner of the food's coordinates

**getWorth**

public int **getWorth**()

Returns the worth of the food

**Returns:**

the worth of the food

**Class Segment**

java.lang.Object

**Segment**

public class **Segment**extends java.lang.Object

The segment class that makes up the Snake.

|  |
| --- |
| **Field Detail** |

**myColor**

public java.awt.Color **myColor**

Color of the segment

**myX**

public int **myX**

Coordinates of the segment

**myY**

public int **myY**

Coordinates of the segment

**mySize**

public int **mySize**

Size of the segment

**myRect**

public java.awt.Rectangle **myRect**

Square with upper left corner of the segment's coordinates

|  |
| --- |
| **Constructor Detail** |

**Segment**

public **Segment**(int x,

int y,

java.awt.Color color)

Segment constructor

**Parameters:**

x - The x-coordinate, in pixels, of myLoc

y - The y-coordinate, in pixels, of myLoc

color - The color of this segment.

|  |
| --- |
| **Method Detail** |

**getColor**

public java.awt.Color **getColor**()

Get the color of the segment

**Returns:**

the current color of the segment

**getX**

public int **getX**()

Get the x-coordinate of the segment

**Returns:**

the x-coordinate of the segment

**getY**

public int **getY**()

Get the y-coordinate of the segment

**Returns:**

the y-coordinate of the segment

**getSize**

public int **getSize**()

Get the size of the segment

**Returns:**

the current size of the segment

**getRect**

public java.awt.Rectangle **getRect**()

Returns the square representing the segment

**Returns:**

the square with upper left corner of the segment's coordinates

**setColor**

public void **setColor**(java.awt.Color c)

Set the color of the segment

**Parameters:**

c - the new color of the segment

**setX**

public void **setX**(int x)

Set the x-coordinate of the segment

**Parameters:**

x - the new x-coordinate of the segment

**setY**

public void **setY**(int y)

Set the y-coordinate of the segment

**Parameters:**

y - the new y-coordinate of the segment

**setSize**

public void **setSize**(int size)

Set the size of the segment

**Parameters:**

size - the new size of the segment

**Class MenuPanel**

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JPanel

**MenuPanel**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

public class **MenuPanel**extends javax.swing.JPanel

MenuPanel that opens when the game runs

|  |
| --- |
| **Field Detail** |

**optPan**

private OptionPanel **optPan**

OptionPanel that opens options screen

**snakePan**

private SnakePanel **snakePan**

SnakePanel that runs game screen

|  |
| --- |
| **Constructor Detail** |

**MenuPanel**

public **MenuPanel**()

Creates new MainPanel with various buttons

|  |
| --- |
| **Method Detail** |

**optButtonActionPerformed**

private void **optButtonActionPerformed**(java.awt.event.ActionEvent evt)

Actions of option button

**snakeButtonActionPerformed**

private void **snakeButtonActionPerformed**(java.awt.event.ActionEvent evt)

Actions of game button

**Class SnakePanel**

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JPanel

**SnakePanel**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

public class **SnakePanel**extends javax.swing.JPanel

The JPanel that runs the main game

**See Also:**

[Serialized Form](serialized-form.html#SnakePanel)

|  |  |
| --- | --- |
| **Nested Class Summary** | |
| class | [**SnakePanel.Key**](SnakePanel.Key.html)           Codes for keyboard input |
| class | [**SnakePanel.Listener**](SnakePanel.Listener.html)           Events that occur every second |
| class | [**SnakePanel.SpeedListener**](SnakePanel.SpeedListener.html)           Moves snake and increments score; change in speed of timer changes speed of snake |

|  |
| --- |
| **Field Detail** |

**FRAME**

private final int **FRAME**

Size of frame

**See Also:**

[Constant Field Values](constant-values.html#SnakePanel.FRAME)

**BACKGROUND**

private final java.awt.Color **BACKGROUND**

Color of background

**myImage**

private java.awt.image.BufferedImage **myImage**

Image that graphics are painted on to

**myBuffer**

private java.awt.Graphics2D **myBuffer**

Graphics class for graphics

**snake**

private Snake **snake**

Main game snake

**mainFood**

private Food **mainFood**

Main game food; can be eaten and regenerated

**lastPressProcessed**

private long **lastPressProcessed**

Keeps the Snake from turning too fast

**isGameOver**

private boolean **isGameOver**

Is the game over?

**wall**

private int **wall**

Size of in-game wall

**score**

private int **score**

Current score

**level**

private int **level**

Current level

**betweenLevels**

private boolean **betweenLevels**

On a screen between levels?

**anyTime**

private long **anyTime**

Used as a stopwatch

**action**

private javax.swing.Timer **action**

Timers for animation and other time-related processes

**snakespeed**

private javax.swing.Timer **snakespeed**

Timers for animation and other time-related processes

|  |
| --- |
| **Constructor Detail** |

**SnakePanel**

public **SnakePanel**()

Constructor of the game panel

|  |
| --- |
| **Method Detail** |

**paintComponent**

public void **paintComponent**(java.awt.Graphics g)

Paint the screen

**Overrides:**

paintComponent in class javax.swing.JComponent

**drawSnake**

public void **drawSnake**()

Draw the snake on the screen

**drawFood**

public void **drawFood**()

Draw the food on the screen

**eatFood**

public void **eatFood**(Food f)

Eat the food, update score, and generate new food

**Parameters:**

f - The Food to be eaten

**gameOver**

public void **gameOver**()

Carries out game over situation

**isDead**

public boolean **isDead**()

Tests if the snake is dead and the game is over

**Returns:**

Is the snake dead?

**increaseLevel**

public void **increaseLevel**()

Increases the level

**Class OptionPanel**

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JPanel

**OptionPanel**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

public class **OptionPanel**extends javax.swing.JPanel

OptionPanel that shows options screen; opened by menu panel. SEE keys FIELD FOR DATA STRUCTURE

**See Also:**

[Serialized Form](serialized-form.html#OptionPanel)

|  |
| --- |
| **Field Detail** |

**keys**

private java.util.Map<java.lang.Integer,java.lang.Integer> **keys**

Map containing all the important actions and the respective keyboard locations

|  |
| --- |
| **Constructor Detail** |

**OptionPanel**

public **OptionPanel**()

Creates new form OptionPanel

|  |
| --- |
| **Method Detail** |

**initComponents**

public void **initComponents**()

Create all the buttons and the GUI of the options screen

**Class Sound**

java.lang.Object

**Sound**

public class **Sound**extends java.lang.Object

A Sound class that plays music and other sounds

|  |
| --- |
| **Field Detail** |

**EXTERNAL\_BUFFER\_SIZE**

private static final int **EXTERNAL\_BUFFER\_SIZE**

Buffer rate of the sound

**See Also:**

[Constant Field Values](constant-values.html#Sound.EXTERNAL_BUFFER_SIZE)

**line**

javax.sound.sampled.Clip **line**

The Clip that plays the sound

**oncePlayed**

boolean **oncePlayed**

Has the sound been played at least once?

|  |
| --- |
| **Constructor Detail** |

**Sound**

public **Sound**(java.lang.String filename)

The constructor of SoundTest. The filename is used to be read in and the audio is later gotten from it and used.

|  |
| --- |
| **Method Detail** |

**play**

public void **play**()

Plays the music.

**loop**

public void **loop**()

Loops the music

**stop**

public void **stop**()

Pauses the music

**exit**

public void **exit**()

Close and exit the file