|  |
| --- |
| Actor |
| - **gamegrid: Gamegrid**  The reference to the GameGrid instance.  - **nbCycles: int**  The current number of simulation cycles since last reset. |
| **Constructors:**  [**Actor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#Actor(boolean, java.lang.String))(boolean isRotatable, java.lang.String filename)  Constructs an actor based on the specified sprite image.  [**Actor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#Actor(java.lang.String))(java.lang.String filename)  Constructs an unrotatable actor based on the specified sprite image. |
| **Methods:**  [**addActorCollisionListener**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#addActorCollisionListener(ch.aplu.jgamegrid.GGActorCollisionListener))(**[GGActorCollisionListener](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/GGActorCollisionListener.html" \o "interface in ch.aplu.jgamegrid)** listener)  Registers a collision listener that reports collision events when actors collide.  [**addCollisionActor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#addCollisionActor(ch.aplu.jgamegrid.Actor))([**Actor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html) partner)  Registers a partner actor that becomes a collision candidate, e.g. that is checked for collisions in every simulation cycle.  [**collide**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#collide(ch.aplu.jgamegrid.Actor, ch.aplu.jgamegrid.Actor))([**Actor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html) actor1, [**Actor**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html) actor2)  Empty implementation of a GGActorCollisionListener called when the two actors collides.  [**getCurrentImage**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getCurrentImage())()  Returns the buffered image of the currently visible sprite picture.  [**getDisplacePosition**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getDisplacePosition())()  Returns the current displace position.  [**getLocation**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getLocation())()  Returns the current location (horizontal and vertical coordinates).  [**isActorCollisionEnabled**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#isActorCollisionEnabled())()  Returns true, if collision notification between actors is enabled.  [**isInGrid**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#isInGrid())()  Returns true, if the actor's location is inside the grid.  [**isMoveValid**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#isMoveValid())()  Returns true, if the next call of move() will put the actor in a cell inside the game grid.  [**setDisplacePosition**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setDisplacePosition(java.awt.geom.Point2D.Double))(java.awt.geom.Point2D.Double displacePosition)  Sets the current displace position and moves the actor in the cell that contains the given coordinates.  [**setLocation**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setLocation(ch.aplu.jgamegrid.Location))([**Location**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Location.html) location)  Assigns a new current location.  [**setLocationOffset**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setLocationOffset(int, int))(int x, int y)  Same as setLoctionOffset(Point locationOffset) with given x and y displacements.  [**setLocationOffset**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setLocationOffset(java.awt.Point))(java.awt.Point locationOffset)  Sets a pixel offset in x- any y-direction relative to the current location.  [**setTileCollisionEnabled**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setTileCollisionEnabled(boolean))(boolean enable)  Enable/disable the detection of collisions with the tile collision candidates.  [**setX**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setX(int))(int x)  Assigns a new current horizontal cell coordinate.  [**setY**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setY(int))(int y)  Assigns a new current vertical cell coordinate.  [**show**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#show())()  Turns on the visibility of the sprite with id 0.  [**show**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#show(int))(int spriteId)  Turns on the visibility of the sprite with given id. |
| Apple | |
| - xLocation: int  - yLocation: int  - score: int / count collisions and add points on score | |
| + [**appleGotEaten**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getLocation())(): int  *addActorCollisionListener()*  *if a collision occurs the apple moves to a random spot on the field*  *using random(), setLocationOffset() and*  **+** [**addActorCollisionListener**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#addActorCollisionListener(ch.aplu.jgamegrid.GGActorCollisionListener))(**[GGActorCollisionListener](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/GGActorCollisionListener.html" \o "interface in ch.aplu.jgamegrid)** listener): void  *Registers a collision listener that reports collision events when actors collide.*  + [**getLocation**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getLocation())(): int  *Returns the current location (horizontal and vertical coordinates).*  + **setLocationOffset**(int x, int y)  *Same as setLoctionOffset(Point locationOffset) with given x and y displacements.*  + [**setX**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setX(int))(int x): void  *Assigns a new current horizontal cell coordinate.*  + **setY**(int y): void  *Assigns a new current vertical cell coordinate.*  + **show**(int spriteId): void  *Turns on the visibility of the sprite with given id.*  + [**random**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#getLocation())(): int  *Returns a random coordinate in the field (x and y).* | |
|  | |

|  |
| --- |
| Snake |
| - xLocation: int  - yLocation: int  - snakeLength: int  - snakeTail: Tail  - snake: Snake[] |
| **+** [**addActorCollisionListener**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#addActorCollisionListener(ch.aplu.jgamegrid.GGActorCollisionListener))(**[GGActorCollisionListener](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/GGActorCollisionListener.html" \o "interface in ch.aplu.jgamegrid)** listener): void  *Registers a collision listener that reports collision events when actors collide.*  + setDirection(double direction)  *Returns the current location (horizontal and vertical coordinates).*  + [**setX**](http://www.aplu.ch/classdoc/jgamegrid/ch/aplu/jgamegrid/Actor.html#setX(int))(int x): void  *Assigns a new current horizontal cell coordinate.*  + **setY**(int y): void  *Assigns a new current vertical cell coordinate.*  + move(int distance)  *Moves the given distance in the current direction.*  + reset()  Empty method called when the actor is added to the game grid. Reverts the program back to the starting point.  + turn(double angle)  Turns the moving direction by the given angle.  + collide()  Empty implementation of GGActorCollisionListener called when two actors collide(in this case if the snake eats itself)  + getImage()  Returns a buffered image.  + getDirection()  Gets the current direction.  + isInGrid():Boolean  Return true if the actor is located within the grid |

|  |
| --- |
| SnakeMovement – Extended from EventListener |
| - xLocation: int  - yLocation: int  - snake: Snake[] |
| + keyPressed(java.awt.event.KeyEvent evt)  *Event callback method called when a key is pressed*  + move(): void  *Moves to one of 3 possible directions based on keyPressed() and based on current direction*  + getDirection(): void  *Gets the current direction of movement* |
|  |