Class Junk

java.lang.Object
    GameObject
        Junk

```
class Junk
extends GameObject
```

A junk file. Increases the CPU usage as it stays on the screen.

Nested Class Summary

Nested classes/interfaces inherited from class GameObject

GameObject.CollHandler

Field Summary

Fields inherited from class GameObject

accel, bgg, bounds, collHandler, collRectOffset, isDead, lastKinematicsVars, position, sprite, velocity

Constructor Summary

Constructors

Constructor and Description
Junk(java.awt.Rectangle bounds) Creates the junk and gives it a bit of downwards acceleration.

Method Summary

Methods

Modifier and Type	Method and Description
void	**collideWith**(GameObject g) All classes should override this method like so: g.getCollHandler().to(this); This code takes the CollHandler of the other object, and calls the handler appropriate for this object.
void	**cycle**() Increase CPU usage by 0.01 per iteration.
void	**onOutOfBounds**() Keep the file on-screen once it has hit the bottom of its boundary.

Methods inherited from class GameObject

applyAccel, applyVelocity, calculateCollRectFromSprite, confine, confine, decelerate, decelerate, getAccel, getAreaRect, getBounds, getCollHandler, getCollRect, getCollRectOffset, getPosition, getSprite, getVelocity, kill, popKinematicsVars, setAccel, setBounds, setCollHandler, setCollRectOffset, setPosition, setSprite, setVelocity, stashKinematicsVars

Methods inherited from class java.lang.Object

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

Constructor Detail

Junk

```
public Junk(java.awt.Rectangle bounds)
```

Creates the junk and gives it a bit of downwards acceleration.

Parameters:

`bounds` - The boundary of the game that created it.

Method Detail

collideWith

```
public void collideWith(GameObject g)
```

Description copied from class: **GameObject**

All classes should override this method like so: g.getCollHandler().to(this); This code takes the CollHandler of the other object, and calls the handler appropriate for this object. This way, handling collisions with various objects can be handled using overloading rather than e.g. object-identifying properties. The advantage is that the decision of which handler to call can be decided at compile-time. More technically, collision handlers have been implemented through the *visitor design pattern*, where implementations of CollHandler are the visitors. Note that collideWith(g) calls g's

handlers, not this object's.

Specified by:

`collideWith` in class `GameObject`

Parameters:

`g` - The other GameObject.

cycle

```
public void cycle()
```

Increase CPU usage by 0.01 per iteration.

Overrides:

`cycle` in class `GameObject`

onOutOfBounds

```
public void onOutOfBounds()
```

Keep the file on-screen once it has hit the bottom of its boundary.

Overrides:

`onOutOfBounds` in class `GameObject`