Class RecycleBin

java.lang.Object
    GameObject
        RecycleBin

```
 class RecycleBin
extends GameObject
```

The recycling bin that collects stuff.

Nested Class Summary

Nested classes/interfaces inherited from class GameObject

GameObject.CollHandler

Field Summary

Fields

Modifier and Type	Field and Description
private long	**amountCollected** The number of items collected.

Fields inherited from class GameObject

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

Constructor Summary

Constructors

Constructor and Description
RecycleBin(java.awt.Rectangle bounds) The constructor.

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**() Every cycle, decelerates the recycle bin according to how many items have been collected.
long	**getAmountCollected**() Returns the number of items collected.
boolean	**isUsed**() Checks if the bin has collected anything.

Methods inherited from class GameObject

applyAccel, applyVelocity, calculateCollRectFromSprite, confine, confine, decelerate, decelerate, getAccel, getAreaRect, getBounds, getCollHandler, getCollRect, getCollRectOffset, getPosition, getSprite, getVelocity, kill, onOutOfBounds, 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

Field Detail

amountCollected

```
private long amountCollected
```

The number of items collected.

Constructor Detail

RecycleBin

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

The constructor. Sets the sprite to an empty bin, which becomes that of the full bin once something has been collected. RecycleBin has collision handlers for Sysfile and Junk. When the recycle bin collides with a Sysfile, the CPU usage is

increased by 6. When Junk is collected, CPU usage decreases by 5. In either case the object is consumed.

Parameters:

`bounds` -

Method Detail

isUsed

```
public boolean isUsed()
```

Checks if the bin has collected anything.

Returns:

true if the bin has collected any items

cycle

```
public void cycle()
```

Every cycle, decelerates the recycle bin according to how many items have been collected. The higher the amount collected, the slower the deceleration. This is construed as "momentum".

Overrides:

`cycle` in class `GameObject`

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.

getAmountCollected

```
public long getAmountCollected()
```

Returns the number of items collected. Determines the difficulty.

Returns:

the number of items collected by the garbage bin.