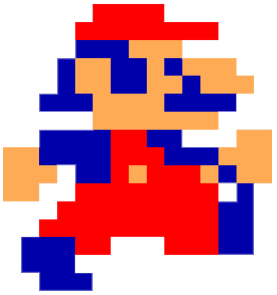
Clase: Personaje



Alto Original:	500px
Ancho Original:	500px
Escala utilizada:	0.090
Alto Utilizado:	40px
Ancho Utilizado:	40px
Distancia centro-suelo:	20px
Distancia centro-lateral:	15px

La clase Personaje permite generar un objeto que representará gráficamente al protagonista del juego.

VARIABLES

Las posiciones en coordenadas X e Y

```
private int posx;
private int posy;
```

Todas las imágenes que forman las animaciones del personaje

```
private Image mirandoIzquierda;
private Image mirandoDerecha;
private Image caminandoIzquierda;
private Image caminandoDerecha;
private Image saltandoIzquierda;
private Image saltandoDerecha;
private Image subiendo;
private Image subiendo_quieto;
```

Esta referencia realizará un aliasing (de las anteriores) a la imagen que debe ser mostrada.

```
private Image imagenMario;
```

Tick en el cual se ejecutó el último salto (o el salto actual)

```
private int tiempoSalto;
```

Indica si está saltando (ascendiendo) o no.

```
private boolean estaSaltando;
```

Indica si está cayendo (es decir que sus pies no están tocando viga alguna.

```
private boolean estaCayendo;
```

Indica si el personaje se encuentra lo suficientemente cerca de una escalera (para poder usarla)

```
private boolean estaCercaEscalera;
```

Indica si el personaje se encuentra dentro (usando) una escalera

```
private boolean estaEnEscalera;
```

Indica el índice que corresponde a la posición de la escalera que se está usando dentro del arreglo de escaleras

```
private int subidoAEscaleraNro;
```

Último archivo de sonido que se usó para caminar, hay 3 variantes.

```
private int sonando;
```

Tick en el cual se ejecutó el último sonido de caminar (ayuda a evitar que suenen sonidos en cada tick)

```
private int sonandoDesde;
```

Esta variable indica si el personaje está mirando a derecha o no (Vital para que se cargue la imagen correcta del personaje según los movimientos que indique el usuario)

```
private boolean miraDerecha;
```

MÉTODOS

- **Constructor** ()

  El constructor asigna:
  - La posición (50,530).
  - Las rutas URL donde se encuentran los archivos GIFs y PNGs
  - Mirando a derecha como verdadero
  - Y los estados “está cayendo”, “está saltando”, “está en una escalera”, y “está cerca de una escalera” como falsos.

- **tocando** (Debe recibir un barril como parámetro)

  Esta función se encarga de comparar las coordenadas de los extremos del personaje y los del barril recibido como parámetro. Luego informa si hay colisión o no.

- **hacerSonar** (Debe recibir el momento actual expresado como un número entero, donde cero es el inicio de ejecución)

  Esta función ejecuta el sonido de caminar pero evita que suene en cada tick donde se está caminando. Sino habría una bola de sonido indistinguible.

  La función decide sonar alguna de las 3 variantes de sonidos de pasos que hay. Y sólo hace sonar cuando la distancia entre el sonido anterior y el actual es de 40 ticks.

- **pisando**(Debe recibir como parámetro, el arreglo con las vigas utilizadas en la instancia Juego)

  Esta función devuelve el índice que ocupa la viga en el arreglo de suelos que el jugador está pisando[1]. Si no se encuentra pisando, entonces devuelve -1.

  Para saber si **no** está pisando la viga, el centro 'y' del personaje + 20 pixeles (para llegar al pie del personaje) **`obtenerPosPies()`** debe poseer un valor distinto para la coordenada 'y' donde comienza cada viga (la posy - 12px) **`(int)suelos[i].dondeEmpiezaElSuelo()`**.

  En el caso de que el personaje se encuentra pisando la viga. Queda por conocer si se encuentra dentro de todos los puntos 'x' que conforman el largo de la viga. Porque de lo contrario, no se encontraría pisándola.

  Por eso la función analiza que el extremo derecho de la viga, sea pisada por al menos el lateral izquierdo del personaje, y lo mismo de forma opuesta.

  Si no se cumple esta condición, el personaje está cayendo por estar fuera de la viga a pesar de estar a la altura de alguna de ellas.

- **estoyCercaDeEscalera** (Debe recibir como parámetro el arreglo con las vigas utilizadas en la instancia Juego; y el arreglo con las escaleras utilizadas en la instancia Juego)

  Esta función cambia el valor de **`estaCercaEscalera`** a true o false dependiendo si el personaje está cerca de una escalera como para poder subir o descender por ella.

  Sólo analiza la proximidad de una escalera, si la función pisando devuelve el índice de la viga pisada. No se analiza proximidad para valores -1 (en el aire) ni si se está cayendo.

  Las escaleras se analizan en dos casos separados. Las que comienzan en el piso actual del personaje y ascienden al superior, y las que terminan en el piso actual porque descienden al inferior. A su vez, dentro de cada caso, se analiza la existencia de una escalera obligatoria y completa, y la de una escalera adicional (sea completa o no).

  Para estar cerca de una escalera los puntos “x” extremos de la escalera deben contener al punto central “x” del personaje. Y la ubicación del punto “y” ocupada por los pies del personaje debe estar a una distancia cercana al extremo correspondiente de la escalera.

[1] Definamos pisando como ocupar el pixel igual o inmediatamente superior del último píxel superior ocupado por una viga. Dicho pìxel ocupado debe ser el primero inferior del personaje.

- **saltandoBarril** (Debe recibir una variable de tipo Barril)

Retorna verdadero cuando se realiza un salto exitoso sobre un barril. Se debe ejecutar en cada tick y se analiza cada barril.

El salto es exitoso cuando:

- La posición “x” del barril es igual a la posición “x” del personaje (con un ayuda de +/- 1 píxel a cada lado)
- Los pies del personaje están por encima de la parte superior del barril
- Pero no a tanta diferencia (tan alto no salta el personaje, sin esta condición los barriles en pisos inferiores serían considerados como saltados)
- Que el barril no haya sido previamente saltado.
- No estar dentro de una escalera.

- **dibujar** (Debe recibir el entorno como parámetro;una variable de tipo entera indicando la rotación de la figura del personaje)

  Dibuja al personaje. Se deben calcular las situaciones y cambiar las variables previamente con otros métodos.

  Utiliza siempre la image almacenada en

- **cambiarImagen** (Debe recibir una variable de tipo String)

  Dependiendo el String recibido, cambia la imagen a la que apunta `imagenMario`. También utiliza para decidir la imagen correcta el valor de `miraDerecha`.

- **obtenerPosPies**(No requiere parámetros adicionales)

  Devuelve la posición del extremo inferior del personaje.

- **obtenerPoscabeza** (No requiere parámetros adicionales)

  Devuelve la posición del extremo superior del personaje.

- **lateralDerecho** (No requiere parámetros adicionales)

  Devuelve la posición del extremo derecho del personaje.

- **lateralIzquierdo** (No requiere parámetros adicionales)

  Devuelve la posición del extremo izquierdo del personaje.

- **estaEnEscalera**(No requiere parámetros adicionales)

  Devuelve verdadero o falso según si está en escalera o no.

- **cambiarY** (Requiere un entero)

  Setea a la posicicón “Y” actual como la suma, al valor actual, del entero recibido.

- **cambiarX**(Requiere un entero)
  Setea a la posicicón “X” actual como la suma, al valor actual, del entero recibido.

  El resto de los métodos son del tipo “**obtener<NombreVariable>**” (get) y “**cambiar<NombreVariable>**” (set).

```
/*
 * Getters
 */

public boolean obtenerEstaEnEscalera() {
	return this.estaEnEscalera;
}

public int obtenerMomentoDeSalto() {
	return this.tiempoSalto;
}

public boolean obtenerEstaCayendo() {
	return this.estaCayendo;
}

public boolean obtenerEstaCercaEscalera() {
	return this.estaCercaEscalera;
}
```

```java
public int obtenerSubidoAEscaleraNro() {
	return this.subidoAEscaleraNro;
}

public boolean obtenerMiraDerecha() {
	return this.miraDerecha;
}

public boolean obtenerEstaSaltando() {
	return this.estaSaltando;
}
```

```java
/*
 * Setters
 */

public void cambiarMomentoDeSalto(int i) {
	this.tiempoSalto = i;

}

public void cambiarEstaEnEscalera(boolean escalera) {
	this.estaEnEscalera = escalera;
}

public void cambiarMiraDerecha(boolean mira) {
	this.miraDerecha = mira;
}

public void cambiarEstaSaltando(boolean salta) {
	this.estaSaltando = salta;
}
```