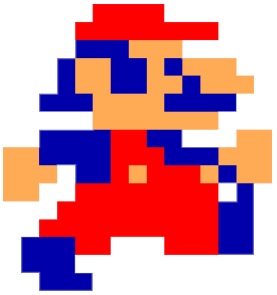
Clase: Personaje



```
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 se encargará de todas las tareas relacionadas al protagonista del juego. Desde mostrar su apartado gráfico, desplazarse por el escenario según lo indique el usuario y las posibilidades en las que se encuentre en determinado momento y a lo largo del tiempo.

VARIABLES

```
private String estado;
```
Indica si el jugador está vivo o muerto. Permite o impide a las funciones que controlan los movimientos del personaje actuar en consecuencia.

```
private int posx;
private int posy;
```
La posición con respecto al eje “x” y al eje “y”.

```
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;
```
Las variables que contienen las animaciones e imágenes en formato GIF o PNG para ilustrar las diferentes acciones del protagonista.

```
private char ultima;
```
Última tecla de sentido (DER o IZQ) presionada (Sirve para saber para dónde debe mirar el personaje).

```
private int tiempoSalto;
```
Tick en el cual se ejecutó el último salto (o el actual)

```
private boolean estaSaltando;
```
Indica si está saltando (ascendiendo) o no.

```
private boolean estaCayendo;
```
Indica si está cayendo (es decir que sus pies no están no están tocando viga

```
                                                    // alguna.
```

```
private boolean estaCercaEscalera;
private boolean estaEnEscalera;
private int enEscalera;

private int sonando = 1;
```

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

```
private int sonandoDesde = 0;
```

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

MÉTODOS *

[1]

- **Constructor** (Requiere que se le pasé un parámetro de tipo Viga)

  El constructor asigna:
  - El estado a “vivo”.
  - La posición x en 50.
  - La posición y 35 píxeles más arriba que el suelo de la Viga del argumento.
  - Las rutas URL donde se encuentran los archivos GIFs y PNGs
  - A la última tecla presionada como la tecla derecha para que mire hacia el lado conveniente.
  - Y los estados “está cayendo”, “está saltando”, “está en una escalera”, y “está cerca de una escalera” como falsos.

- **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.

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

  La función saltar se encarga de la parte del salto que se ejecuta una sola vez. Es decir que **no** se encarga de la animación de subida o caída a lo largo de los ticks de un salto normal.

  Se le debe indicar el entorno y el contador de ticks actual.

[1] (*) A excepción que se indique lo contrario, los métodos reciben una variable de tipo **Entorno** y que debe ser la instancia creada por la clase **Juego**.

Cambia el dibujo de caminar por el salto, según hacia qué lado este mirando el personaje. Cambia el estado de **`estaSaltando`** a verdadero. Ejecuta el sonido del salto. Indica el tick en el cual se realizó el salto, guardando el valor en **`tiempoSalto`**.

- **saltando** (Debe recibir el entorno como parámetro; el momento actual expresado como un número entero, donde cero es el inicio de ejecución; y el arreglo con las vigas utilizadas en la instancia Juego)

  Esta función se encarga de manipular, a lo largo del tiempo, lo que ocurre con el personaje cuando no está en el suelo (si debe ascender porque saltó, sí debe caer o ninguna de las anteriores si ya se encuentra en el suelo). Para que funcione correctamente se la llamar por cada tick.

  Si el personaje está vivo y no se encuentra desplazándose en una escalera, se evalúa lo siguiente:

  - Si el personaje está saltando, y dicho salto se produjo con menos de 30 ticks de diferencia con el momento actual, entonces hay que elevar 1px al jugador (restar 1 en eje 'y').
  - En cualquier otro caso, se indica que el personaje ya no está saltando y se consulta al método **`pisando`**.
    Si el método **`pisando`** informa que no se está pisando ninguna viga (valor -1) el jugador debe caer (aumentar 1 en el eje “y”) hasta que sus pies toquen suelo. Si por contrario, **`pisando`** informa que se está exáctamente sobre una viga (devuelve el índice de la viga pisada).

- **pisando**(Debe recibir el entorno como parámetro y 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[2]. 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) **`pies()`** 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.

[2] 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.

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

  Devuelve la posición del extremo inferior del personaje.

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

  Devuelve la posición del extremo superior del personaje.

- **dibujar**(Debe recibir el entorno como parámetro; el momento actual expresado como un número entero, donde cero es el inicio de ejecución; y el arreglo con las escaleras utilizadas en la instancia Juego)

  Este método se encarga de ejecutar todas las animaciones y de desplazar (u ordenar a otra función desplazar) al personaje según se cumplan los criterios que lo permitan.

  En primer lugar la función analiza si el estado del personaje es “vivo”. De lo contrario lo hace caer hasta quitarlo de pantalla. Esto es cuando el jugador pierde.

  Para el estado “vivo” el método analiza diferentes condiciones para comprender cómo se debe actuar:

  - Si se presiona la tecla espacio (saltar), ordenará ejecutar la función **`saltar`**. Pero sólo si a su vez se cumple lo siguiente: La distancia temporal del momento actual con respecto al salto anterior (**`tiempoSalto`**) debe ser superior a 60 ticks; no se debe estar cayendo; no se debe estar desplazando dentro de una escalera.

  - Si se presiona la tecla arriba o abajo, ordenará desplazarse por una escalera en la dirección solicitada. Pero sólo si a su vez se cumple lo siguiente: Se debe estar cerca de una escalera.
    - Este apartado a su vez controla el ingresar a una escalera. Puesto que si la escalera comunica el piso actual con el inferior, la única forma de ingresar a la escalera, es hacia abajo. Si la escalera comunica el piso actual con el superior, es hacia arriba.
    - El desplazamiento por una escalera donde el personaje ya se encuentre.
    - La imágen estática del personaje si decide quedarse quieto en la escalera.

  - Si se presiona las teclas izquierda o derecha, mostrará la animación correspondiente y desplazará al jugador. Pero sólo si a su vez se cumple lo siguiente: No se debe estar cayendo, no se debe estar saltando, no se debe estar dentro de una escalera.

- **estoyCercaDeEscalera** (Debe recibir el entorno 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.

  Esta función debe llamarse en cada tick del juego pero sólo si el personaje no se encuentra dentro de una escalera actualmente.

  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.

  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.

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

  Esta función ejecuta las animaciones correspondiente a subir escalera y se encarga de informar si ya terminó de subirla. Es decir que sale de la escalera y se encuentra en el piso superior.

  Según si el nuevo piso al que se ascendió, la escalera se encontrara a derecha o izquierda, voltea al personaje en la dirección correcta.

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

  Esta función ejecuta las animaciones correspondiente a bajar escalera y se encarga de informar si ya terminó de descender. Es decir que sale de la escalera y se encuentra en el piso inferior.

  Según si el piso al que descendió, la escalera se encontrara a derecha o izquierda, voltea al personaje en la dirección correcta.

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

  Devuelve la posición del extremo derecho del personaje.

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

  Devuelve la posición del extremo izquierdodel personaje.

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

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

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

  Cambia el estado a “muerto”.

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

  Retorna verdadero sólo cuando el jugador se encuentra en una posición x igual o menor a 150 y a la vez en la última viga del arreglo (donde se encuentra donkey).

- **saltandoBarril** (Debe recibir el arreglo con los barriles utilizadas en la instancia Juego)

  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.