Programación 2

Copia, clon e igualdad superficial y en profundidad

Repaso - Estado interno

El **estado interno** de un objeto queda determinado por los atributos de instancia de su clase.

Una variable de tipo clase está **ligada** si mantiene una **referencia** al **estado interno de un objeto**.

El acceso a los atributos del objeto debe restringirse de forma que solamente el objeto pueda acceder directamente a sus atributos. Si otro objeto necesita acceder a los valores del estado interno debe hacerlo con los servicios que ofrece la clase.

Repaso - Identidad, igualdad y equivalencia

Cada objeto de software tiene una **identidad**, una **propiedad** que lo distingue de los demás.

La **referencia** a un objeto puede ser usada como propiedad para identificarlo.

Si dos variables son iguales, mantienen una misma referencia, entonces están ligadas a un mismo objeto.

Cuando dos objetos mantienen el mismo estado interno, decimos que son **equivalentes**, aun cuando tienen diferente identidad.

Repaso - Identidad, igualdad y equivalencia (superficial)

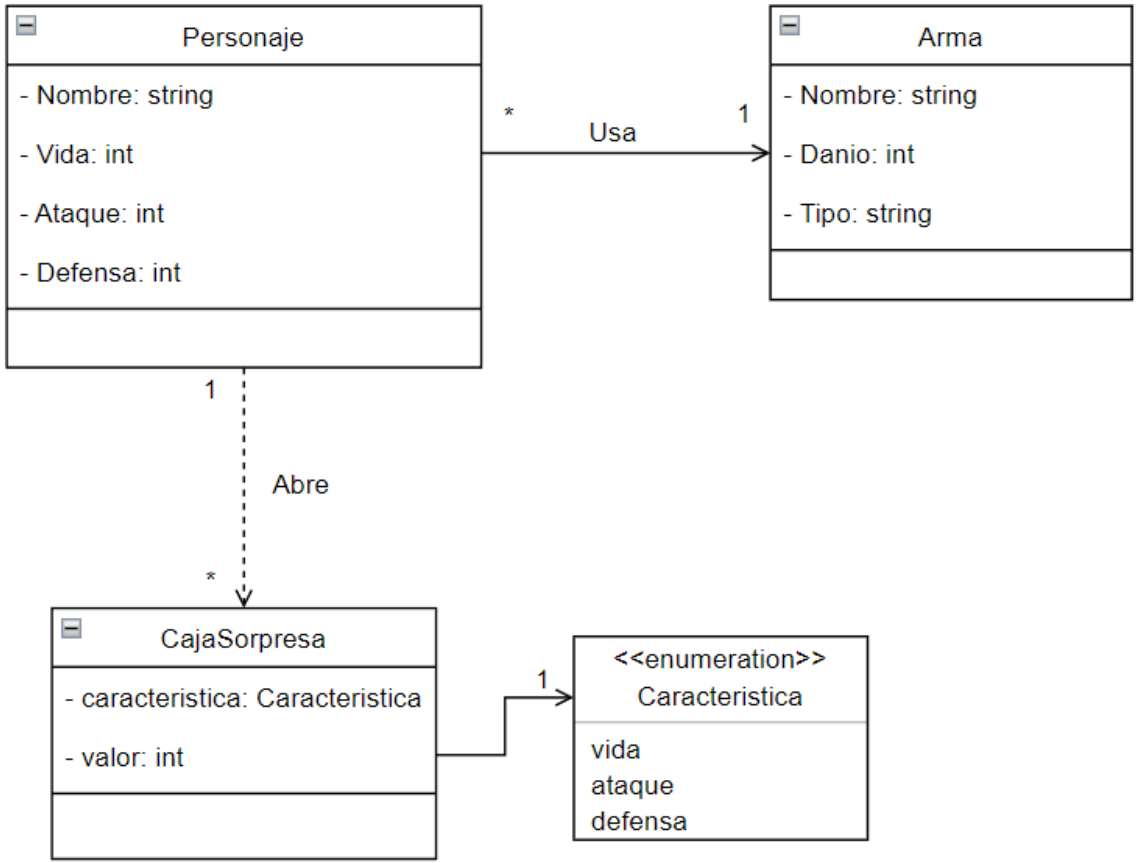
Es una copia de un objeto en la que sólo se duplican las referencias a los objetos contenidos, en lugar de duplicar los propios objetos. Es decir, si el objeto original contiene referencias a otros objetos, la copia superficial no crea nuevos objetos, sino que copia esas referencias.

Esto implica que cualquier modificación en los objetos referenciados por el clon se reflejará en el objeto original, ya que ambos apuntan a los mismos datos internos.

Caso de estudio: Juego de rol simple

Retomemos el caso de estudio *Juego de Rol* .

Nos centraremos principalmente en las clases asociadas (personaje - arma)



Caso de estudio: Juego de rol simple

Personaje

<<atributos de clase>>
- max_vida, max_ataque, max_defensa, min_vida, min_ataque, min_defensa: int

<<atributos de instancia>>
- nombre : String
- vida: entero
- ataque: entero
- defensa: entero
- arma: Arma

<<constructor>>
+ Personaje (nombre: string, ataque: entero, defensa: entero)

<<consultas>>
+ estaVivo(): boolean
+ clonar(): Personaje
+ esIgual(otroPersonaje: Personaje): boolean

<<comandos>>
+ atacar(otroPersonaje: Personaje)
+ recibirAtaque(valorAtaque: entero)
+ abrirCaja(caja: CajaSorpresa)
+ copiarValores(otroPersonaje: Personaje)

Arma

<<atributos de instancia>>
- nombre : String
- danio: entero
- tipo: entero

<<constructor>>
+ Arma(nombre: string, tipo: string, danio: entero)

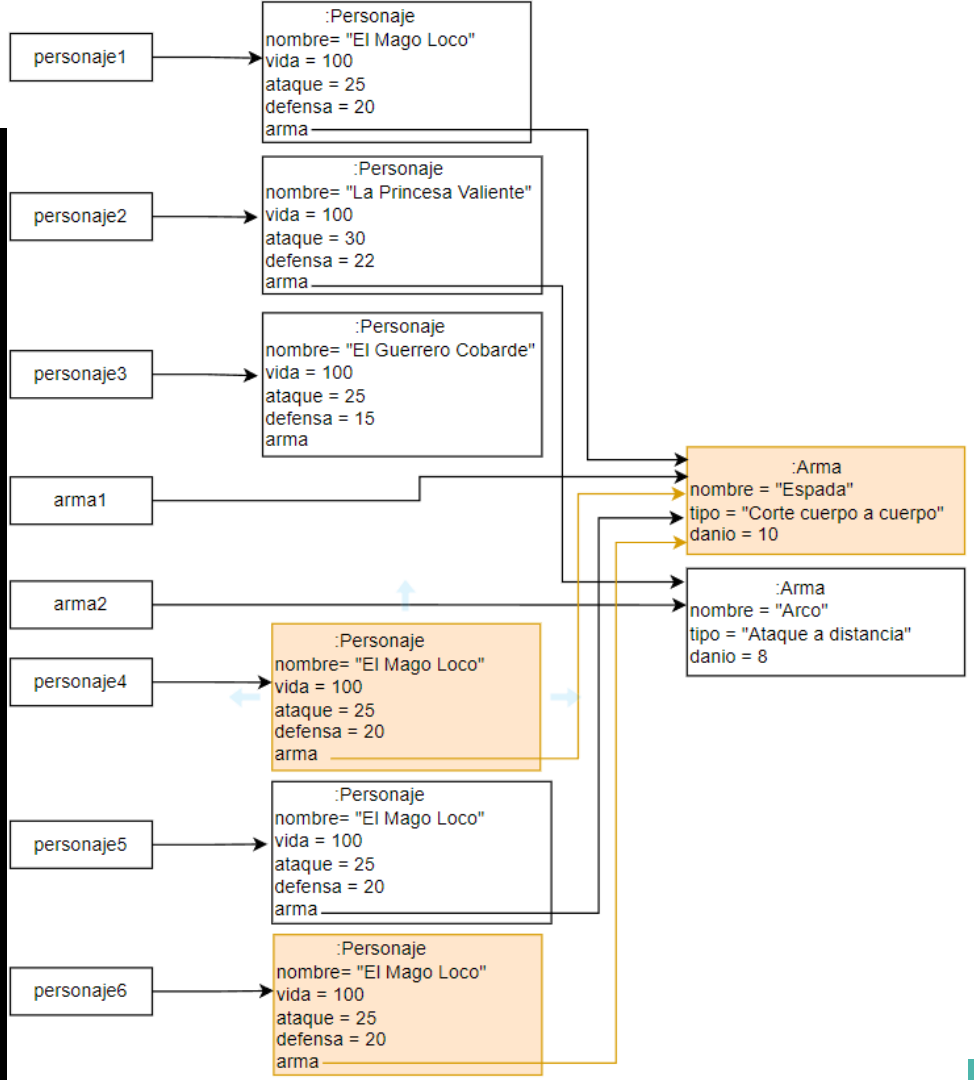
<<consultas>>
+ obtenerNombre(): string
+ obtenerTipo(): string
+ obtenerDanio(): entero
+ toString(): string

Caso de estudio: Juego de rol simple

En la clase TesterPersonaje habíamos observado que los comandos `clonar(), copiarValores()` y `esIgual()` efectuaban una copia o evaluación de los valores elementales contenidos y las referencias a los objetos contenidos, en lugar de duplicar los propios objetos o evaluar los estados internos de los objetos referenciados, como podemos observar a continuación:

Diagrama de objetos

```python
[ ... imports ... ]
class TesterPersonaje:
    @staticmethod
    def test():
        separador = "-"*70
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        [ ... ]
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        [ ... ]
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)

#prueba los metodos clonar(), esIgual() y copiarValores()
        personaje4 = personaje1.clonar()
        print(personaje4)
        personaje5 = personaje2.clonar()
        personaje5.copiarValores(personaje4)
        personaje6=personaje1.clonar()
        esIgual = personaje6.esIgual(personaje4)
```

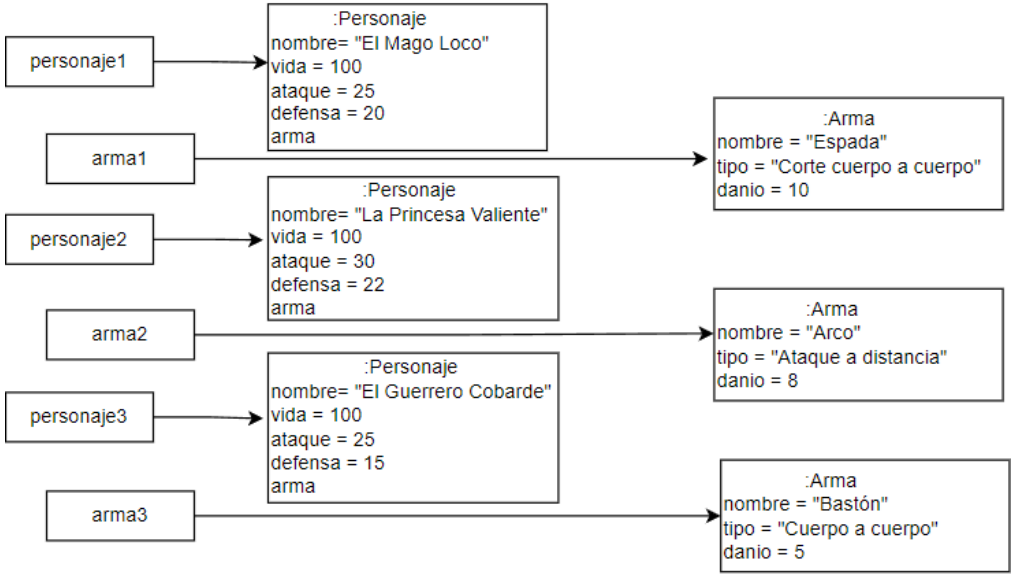


Caso de estudio: Juego de rol simple

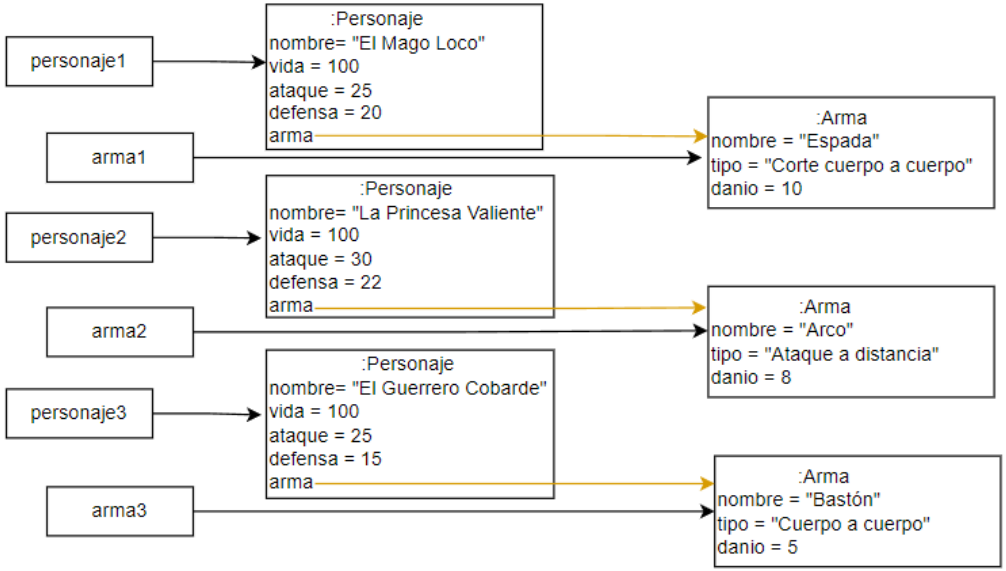
Vamos a replantear la clase tester y evaluemos los resultados paso a paso:

```
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```

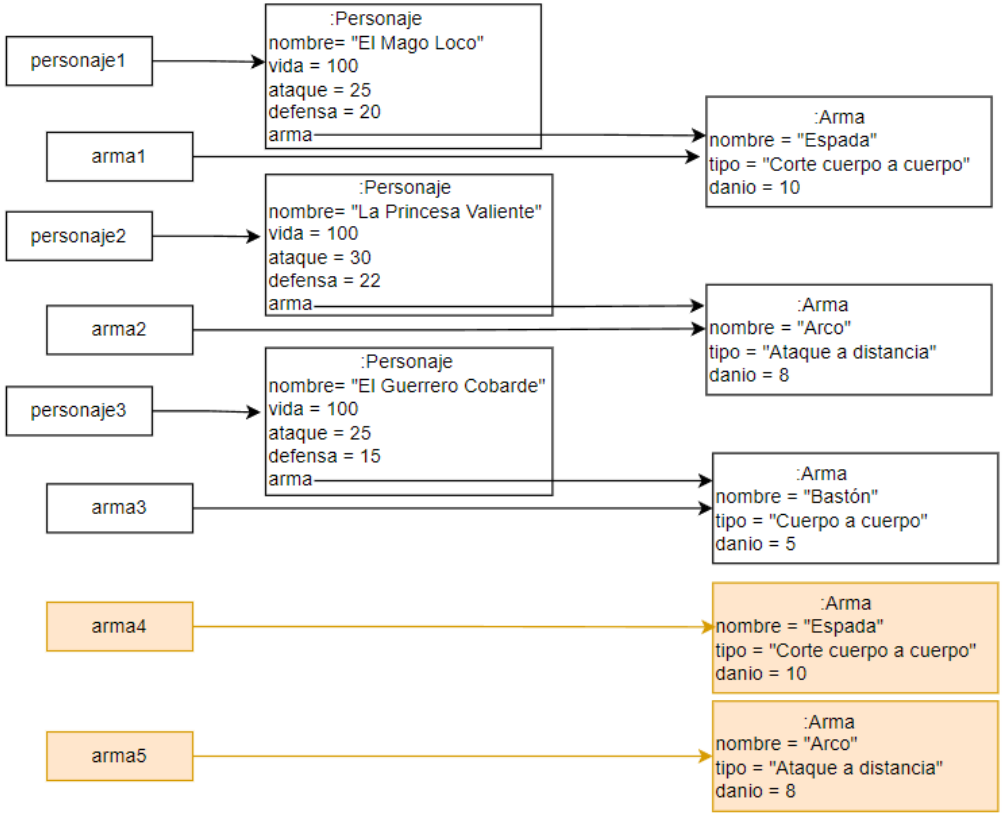
```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```



```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```



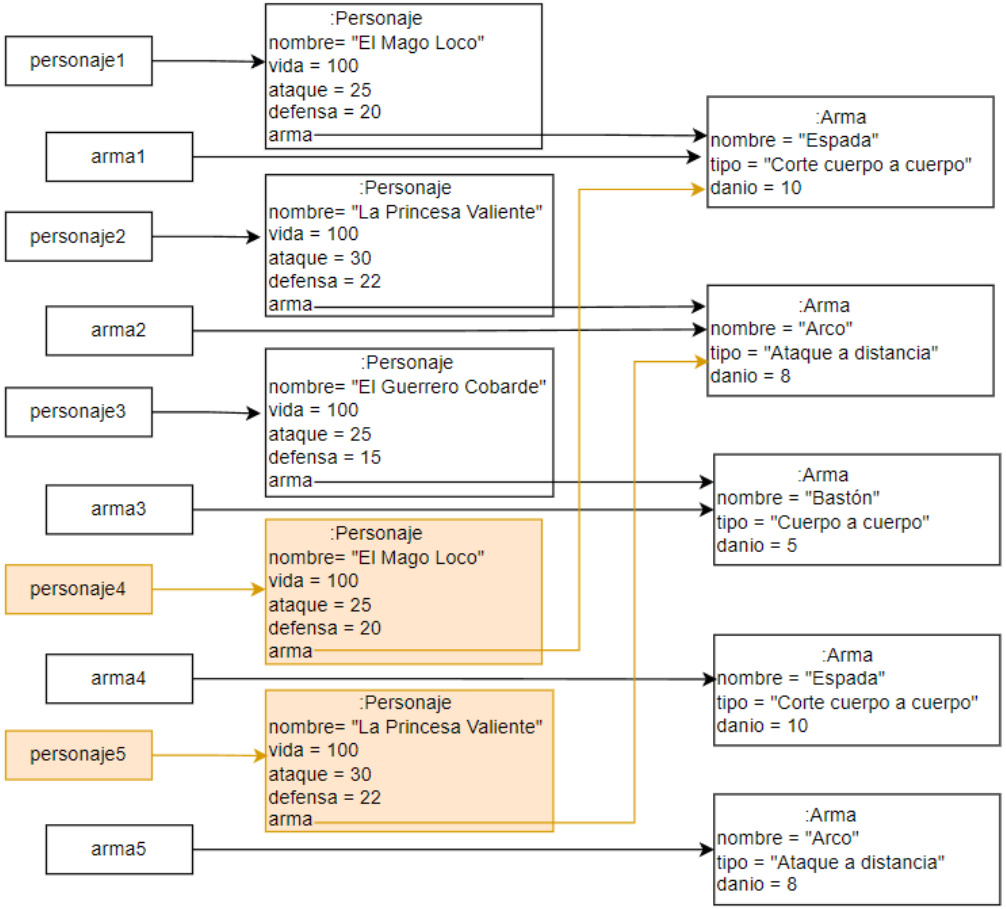
```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```



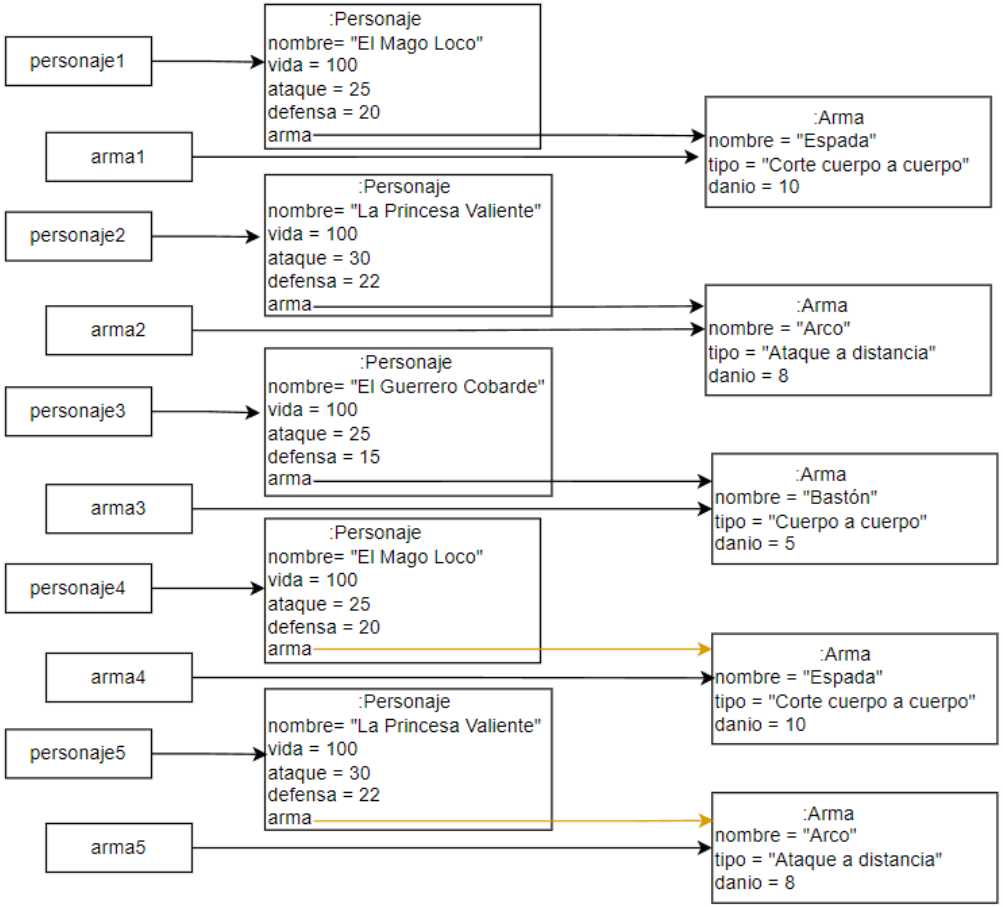
```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```

```python
class Personaje:

[ . . . ]

def clonar(self)->"Personaje":
    """Devuelve un clon del personaje."""
    clon = Personaje(self.__nombre, self.__ataque, self.__defensa)
    clon.establecerVida(self.__vida)
    clon.establecerArma(self.__arma)
    return clon
```

```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```



```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```



```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")

        print("personaje2 -> ", personaje2)
        print("personaje5 -> ", personaje5)
        if personaje2.esIgual(personaje5):
            print("Los personajes 2 y 5 son iguales.")
        else:
            print("Los personajes 2 y 5 son distintos.")
```

```python
class Personaje:

[ . . . ]

def esIgual(self, otro:"Personaje")->bool:
    """Devuelve True si el personaje es igual a otro, False en caso contrario.
Retorna ValueError si otro no es un objeto de la clase Personaje."""
    if isinstance(otro, Personaje):
        return self.__nombre==otro.obtenerNombre() and
self.__vida==otro.obtenerVida() and self.__ataque==otro.obtenerAtaque() and
self.__defensa==otro.obtenerDefensa() and self.__arma==otro.obtenerArma()
    else:
        raise ValueError("El personaje a comparar debe ser un objeto de la clase
Personaje.")
```

Salida: ??

personaje1 -> Nombre: El Mago Loco, Vida: 100, Ataque: 25, Defensa: 20, Arma: Espada - Corte cuerpo a cuerpo (+10 de daño)

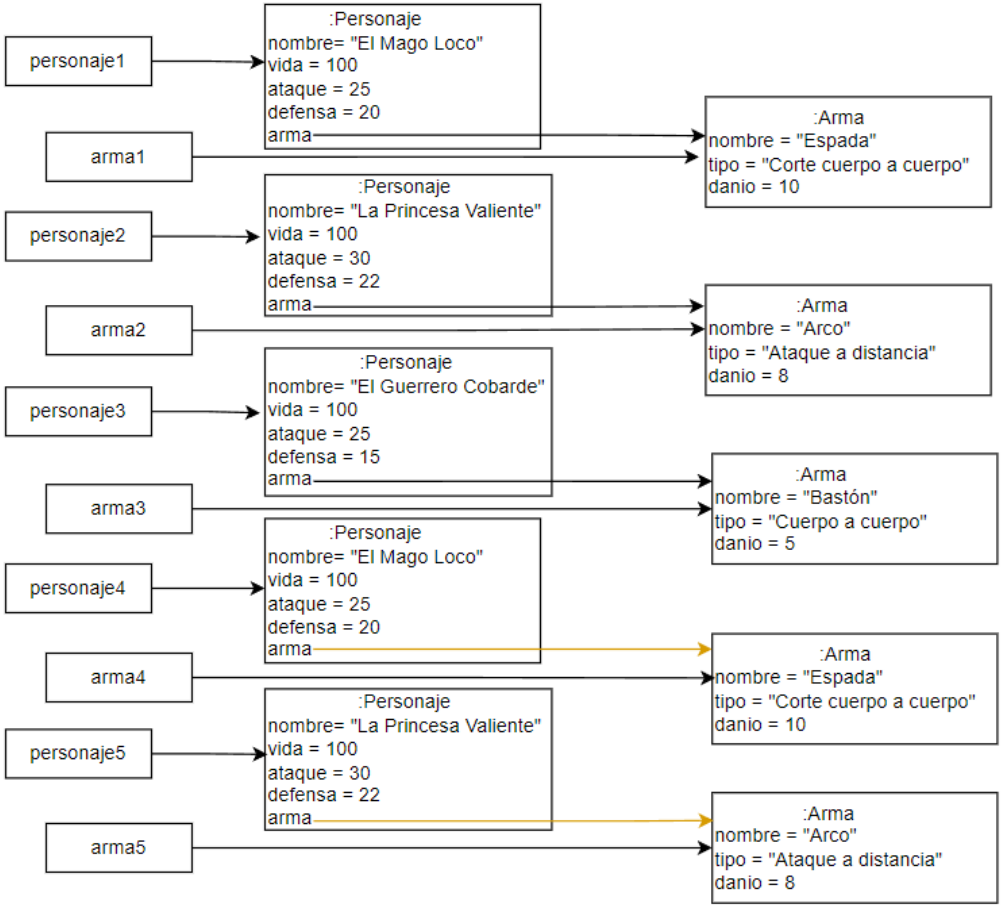
personaje4 -> Nombre: El Mago Loco, Vida: 100, Ataque: 25, Defensa: 20, Arma: Espada - Corte cuerpo a cuerpo (+10 de daño)

Los personajes 1 y 4 son distintos.

¿Por qué?

El método esIgual() en Personaje fue implementado de forma superficial, al igual que los métodos clonar() y copiarValores()

A pesar de que Personaje1 y Personaje4 posean armas que son equivalentes y los mismos valores para el resto de sus atributos, al no mantener las mismas referencias en “`self.__arma`” no son equivalentes entre sí.



Como venimos?

Copia en profundidad o Copia Profunda

En programación orientada a objetos, una **copia profunda** (o deep copy) implica la duplicación completa de un objeto, junto con todos los objetos a los que este hace referencia, de forma recursiva. Esto significa que tanto el objeto original como sus dependencias son replicados en nuevas instancias, de modo que el **clon** sea **completamente independiente del objeto original**.

Es decir, se crea un nuevo objeto con el mismo estado interno que el original, y si ese objeto contiene referencias a otros objetos, también se crean nuevas instancias de esos objetos referenciados. El resultado es que el clon es una réplica exacta del objeto original, pero cualquier cambio hecho en el clon no afecta al objeto original, y viceversa.

La copia profunda es útil cuando queremos asegurarnos de que un objeto sea completamente independiente del original, especialmente cuando se trata de objetos que contienen otros objetos complejos. Al modificar cualquier parte del clon, no afectamos al original, ya que no comparten ninguna referencia interna.

Copia en profundidad vs Copia Superficial

En una copia superficial, se duplica el objeto principal, pero no los objetos referenciados dentro de él. Por lo tanto, los objetos referenciados (como listas, diccionarios, o instancias de otras clases) son compartidos entre el original y la copia, entonces los cambios efectuados en uno se ven reflejados en el otro.

En la copia profunda, tanto el objeto principal como los objetos referenciados son duplicados. Esto asegura que cualquier cambio en la copia no afecte al original, ya que los dos objetos son completamente independientes.

Personaje

<<atributos de clase>>
- max_vida, max_ataque, max_defensa, min_vida, min_ataque, min_defensa: int

<<atributos de instancia>>
- nombre : String
- vida: entero
- ataque: entero
- defensa: entero
- arma: Arma

<<constructor>>
+ Personaje (nombre: string, ataque: entero, defensa: entero)

<<consultas>>
+ estaVivo(): boolean
+ clonarSup(): Personaje
+ clonarProf(): Personaje
+ esIgualSup(otroPersonaje: Personaje): boolean
+ esIgualProf(otroPersonaje: Personaje): boolean

<<comandos>>
+ atacar(otroPersonaje: Personaje)
+ recibirAtaque(valorAtaque: entero)
+ abrirCaja(caja: CajaSorpresa)
+ copiarValoresSup(otroPersonaje: Personaje)
+ copiarValoresProf(otroPersonaje: Personaje)

clonarSup() devuelve un nuevo objeto creado con los mismos atributos internos y las mismas referencias contenidas

clonarProf() devuelve un nuevo objeto creado con los mismos atributos internos y con copias independientes de los objetos referenciados.

esIgualSup(otroPersonaje: Personaje): evalúa superficialmente el estado interno del personaje recibido por parámetro con el estado interno de la instancia que recibe el mensaje, si son equivalentes devuelve True

esIgualProf(otroPersonaje: Personaje): evalúa en profundidad el estado interno del personaje recibido por parámetro con el estado interno de la instancia que recibe el mensaje, si sus atributos son iguales y los objetos referenciados son equivalentes, devuelve True

copiarValoresSup(otroPersonaje: Personaje) actualiza el estado interno del personaje que recibe el mensaje con los valores y referencias contenidas en el estado interno del personaje recibido por parámetro.

copiarValoresProf(otroPersonaje: Personaje) actualiza el estado interno del personaje que recibe el mensaje con los valores y objetos equivalentes del estado interno del personaje recibido por parámetro.

Para evaluar en profundidad Personaje, primero debemos adaptar la clase Arma para que ofrezca los servicios que necesitaremos

Arma
<<atributos de instancia>> - nombre : String - danio: entero - tipo: entero
<<constructor>> + Arma(nombre: string, tipo: string, danio: entero) **<<consultas>>** + obtenerNombre(): string + obtenerTipo(): string + obtenerDanio(): entero + toString(): string **+ clonar(): Arma** **+ esIgual(otraArma: Arma): boolean**

```python
def clonar(self):
    return Arma(self.__nombre, self.__tipo, self.__danio)


def esIgual(self, otra_arma:"Arma")->bool:
    """Determina si una arma es igual a otra. Si no recibe un objeto de tipo Arma lanza un ValueError."""
    if not isinstance(otra_arma, Arma):
        raise ValueError("El objeto a comparar debe ser de tipo Arma.")
    return self.__nombre == otra_arma.obtenerNombre() and self.__tipo == otra_arma.obtenerTipo() and self.__danio == otra_arma.obtenerDanio()
```

Personaje

<<atributos de clase>>
- max_vida, max_ataque, max_defensa, min_vida, min_ataque, min_defensa: int

<<atributos de instancia>>
- nombre : String
- vida: entero
- ataque: entero
- defensa: entero
- arma: Arma

<<constructor>>
+ Personaje (nombre: string, ataque: entero, defensa: entero)

<<consultas>>
+ estaVivo(): boolean
+ clonarSup(): Personaje
+ clonarProf(): Personaje
+ esIgualSup(otroPersonaje: Personaje): boolean
+ esIgualProf(otroPersonaje: Personaje): boolean

<<comandos>>
+ atacar(otroPersonaje: Personaje)
+ recibirAtaque(valorAtaque: entero)
+ abrirCaja(caja: CajaSorpresa)
+ copiarValoresSup(otroPersonaje: Personaje)
+ copiarValoresProf(otroPersonaje: Personaje)

```python
#clonación en profundidad
def clonarSup(self)->"Personaje":
    """Devuelve un clon del personaje."""
    clon = Personaje(self.__nombre, self.__ataque, self.__defensa)
    clon.establecerVida(self.__vida)
    clon.establecerArma(self.__arma)
    return clon

#clonación en profundidad
def clonarProf(self)->"Personaje":
    """Devuelve un clon en profundidad del personaje."""
    clon = Personaje(self.__nombre, self.__ataque, self.__defensa)
    clon.establecerVida(self.__vida)
    if self.__arma!=None:
        clon.establecerArma(self.__arma.clonar())
    return clon
```

La consulta `clonar()` crea y retorna como resultado un objeto con el mismo estado interno que el objeto que recibe el mensaje.

- En la clonación superficial (*clonarSup()*) los dos personajes quedan asociados a un mismo arma (misma referencia).
- En la clonación en profundidad (*clonarProf()*) el nuevo personaje queda asociado a un nuevo arma, que es un clon del arma del personaje que recibe el mensaje. Así, las armas de los dos personajes tienen distinta identidad pero son equivalentes.

Personaje

<<atributos de clase>>
- max_vida, max_ataque, max_defensa, min_vida, min_ataque, min_defensa: int

<<atributos de instancia>>
- nombre : String
- vida: entero
- ataque: entero
- defensa: entero
- arma: Arma

<<constructor>>
+ Personaje (nombre: string, ataque: entero, defensa: entero)

<<consultas>>
+ estaVivo(): boolean
+ clonarSup(): Personaje
+ clonarProf(): Personaje
+ esIgualSup(otroPersonaje: Personaje): boolean
+ esIgualProf(otroPersonaje: Personaje): boolean

<<comandos>>
+ atacar(otroPersonaje: Personaje)
+ recibirAtaque(valorAtaque: entero)
+ abrirCaja(caja: CajaSorpresa)
+ copiarValoresSup(otroPersonaje: Personaje)
+ copiarValoresProf(otroPersonaje: Personaje)

```python
#igualdad superficial
def esIgualSup(self, otro:"Personaje")->bool:
    """Devuelve True si el personaje es igual a otro, False en caso contrario. Retorna ValueError si otro no es un objeto de la clase Personaje."""
    if isinstance(otro, Personaje):
        return self.__nombre==otro.obtenerNombre() and self.__vida==otro.obtenerVida() and self.__ataque==otro.obtenerAtaque() and self.__defensa==otro.obtenerDefensa() and self.__arma==otro.obtenerArma()
    else:
        raise ValueError("El personaje a comparar debe ser un objeto de la clase Personaje.")


#igualdad en profundidad
def esIgualProf(self, otro:"Personaje")->bool:
    """Devuelve True si el personaje es igual a otro, False en caso contrario. Retorna ValueError si otro no es un objeto de la clase Personaje."""
    if isinstance(otro, Personaje):
        return self.__nombre==otro.obtenerNombre() and self.__vida==otro.obtenerVida() and self.__ataque==otro.obtenerAtaque() and self.__defensa==otro.obtenerDefensa() and self.__arma.esIgual(otro.obtenerArma())
    else:
        raise ValueError("El personaje a comparar debe ser un objeto de la clase Personaje.")
```

La consulta esIgual compara el estado interno del objeto que recibe el mensaje con el estado interno del objeto que pasa como parámetro.

- La **igualdad superficial** (*esIgualSup()*) computa true si los dos personajes tienen el mismo valor en los atributos vida, ataque y defensa, y están asociados a una misma arma, esto es **las referencias son iguales**.
- La **igualdad en profundidad** (*esIgualProf()*) computa true si los dos personajes tienen el mismo valor en los atributos vida, ataque y defensa, y están asociados a **armas equivalentes**.

Personaje

<<atributos de clase>>
- max_vida, max_ataque, max_defensa, min_vida, min_ataque, min_defensa: int

<<atributos de instancia>>
- nombre : String
- vida: entero
- ataque: entero
- defensa: entero
- arma: Arma

<<constructor>>
+ Personaje (nombre: string, ataque: entero, defensa: entero)

<<consultas>>
+ estaVivo(): boolean
+ clonarSup(): Personaje
+ clonarProf(): Personaje
+ esIgualSup(otroPersonaje: Personaje): boolean
+ esIgualProf(otroPersonaje: Personaje): boolean

<<comandos>>
+ atacar(otroPersonaje: Personaje)
+ recibirAtaque(valorAtaque: entero)
+ abrirCaja(caja: CajaSorpresa)
+ copiarValoresSup(otroPersonaje: Personaje)
+ copiarValoresProf(otroPersonaje: Personaje)

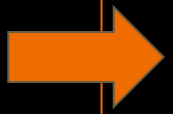
```python
#copia superficial
def copiarValoresSup(self, otro:"Personaje"):
    """Copia los valores de otro personaje. Si otro no es un objeto de la clase Personaje, lanza un ValueError."""
    if isinstance(otro, Personaje):
        self.__nombre = otro.obtenerNombre()
        self.__vida = otro.obtenerVida()
        self.__ataque = otro.obtenerAtaque()
        self.__defensa = otro.obtenerDefensa()
        self.__arma = otro.obtenerArma()
    else:
        raise ValueError("El personaje a copiar debe ser un objeto de la clase Personaje.")

#copia en profundidad
def copiarValoresProf(self, otro:"Personaje"):
    """Copia en profundidad los valores de otro personaje. Si otro no es un objeto de la clase Personaje, lanza un ValueError."""
    if isinstance(otro, Personaje):
        self.__nombre = otro.obtenerNombre()
        self.__vida = otro.obtenerVida()
        self.__ataque = otro.obtenerAtaque()
        self.__defensa = otro.obtenerDefensa()
        if otro.obtenerArma()!=None:
            self.__arma = otro.obtenerArma().clonar()
    else:
        raise ValueError("El personaje a copiar debe ser un objeto de la clase Personaje.")
```

El comando copiarValores modifica el estado interno del objeto que recibe el mensaje con el estado interno del objeto que pasa como parámetro.

- En la **copia superficial** (*copiarValoresSup()*) los dos personajes quedan asociados a un mismo arma. La referencia al arma del personaje que pasa como parámetro, se asigna al atributo de instancia __arma del personaje que recibe el mensaje.
- En la **copia en profundidad** (*copiarValoresProf()*) los dos personajes quedan asociados a distintas armas, equivalentes entre sí. El estado interno del arma del personaje que pasa como parámetro, se copia en el estado interno del arma asociada al personaje que recibe el mensaje.

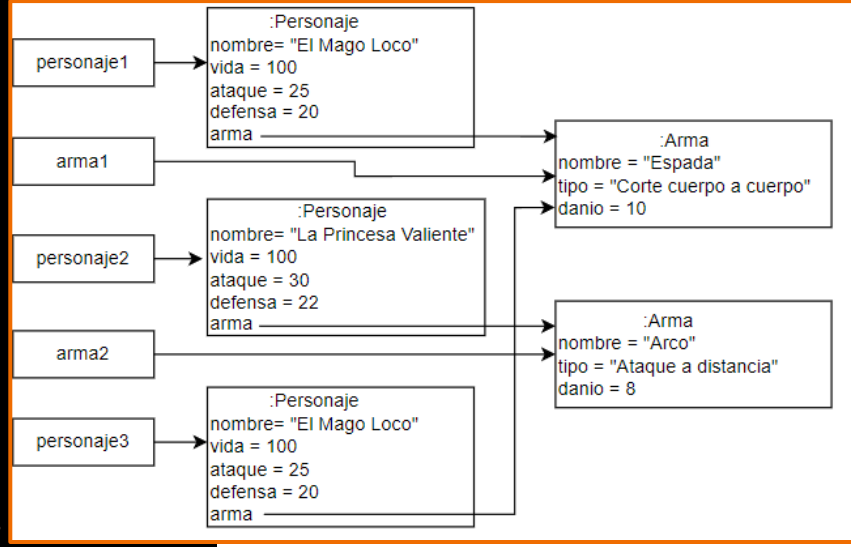
Modifiquemos un poco la secuencia del test para ver en funcionamiento la copia superficial y la copia en profundidad

```python
class TesterValoresCopias:
    @staticmethod
    def test():
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        personaje3 = Personaje("El Guerrero Cobarde", 25, 15)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        arma3 = Arma("Bastón", "Cuerpo a cuerpo", 5)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)
        personaje3.establecerArma(arma3)

        arma4 = arma1.clonar()
        arma5 = arma2.clonar()
        personaje4 = personaje1.clonar()
        personaje5 = personaje2.clonar()
        personaje4.establecerArma(arma4)
        personaje5.establecerArma(arma5)
        print("personaje1 -> ", personaje1)
        print("personaje4 -> ", personaje4)
        if personaje1.esIgual(personaje4):
            print("Los personajes 1 y 4 son iguales.")
        else:
            print("Los personajes 1 y 4 son distintos.")
```

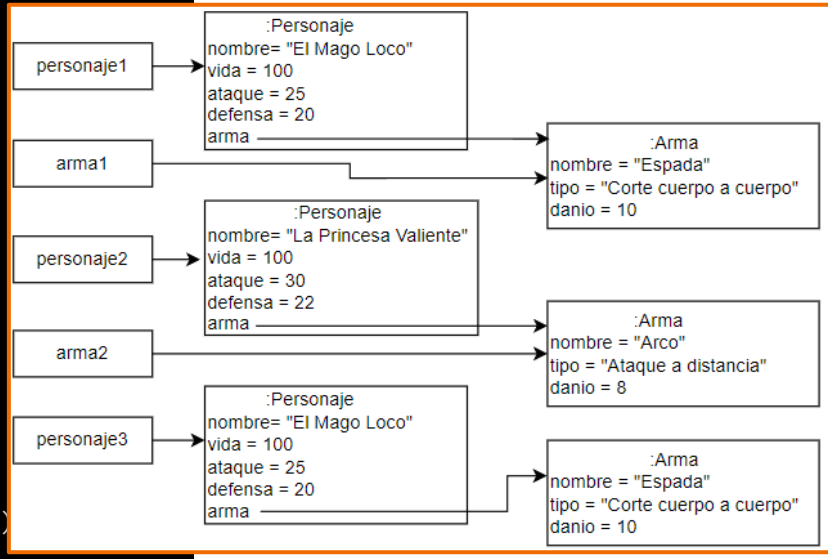


```python
class TesterValoresCopias:
    @staticmethod
    def test():
        separador = "-"*70
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)

        personaje3 = personaje1.clonarSup()
        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")

        personaje3 = personaje1.clonarProf()

        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")
        if personaje1.esIgualProf(personaje3):
            print("Los personajes 1 y 3 son iguales (eval. profunda).")
        else:
            print("Los personajes 1 y 3 son distintos (eval.
```

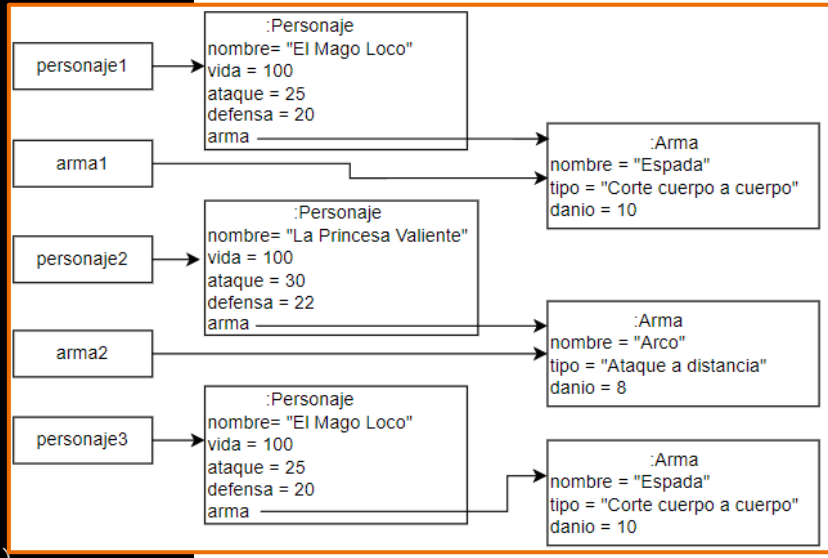
```python
class TesterValoresCopias:
    @staticmethod
    def test():
        separador = "-"*70
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)

        personaje3 = personaje1.clonarSup()
        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")

        personaje3 = personaje1.clonarProf()

        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")
        if personaje1.esIgualProf(personaje3):
            print("Los personajes 1 y 3 son iguales (eval. profunda).")
        else:
            print("Los personajes 1 y 3 son distintos (eval. profunda).")
```



```python
class TesterValoresCopias:
    @staticmethod
    def test():
        separador = "-"*70
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)

        personaje3 = personaje1.clonarSup()
        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")

        personaje3 = personaje1.clonarProf()

        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")
        if personaje1.esIgualProf(personaje3):
            print("Los personajes 1 y 3 son iguales (eval. profunda).")
        else:
            print("Los personajes 1 y 3 son distintos (eval. profunda).")
```



```python
class TesterValoresCopias:
    @staticmethod
    def test():
        separador = "-"*70
        personaje1 = Personaje("El Mago Loco", 25, 20)
        personaje2 = Personaje("La Princesa Valiente", 30, 22)
        arma1 = Arma("Espada", "Corte cuerpo a cuerpo", 10)
        arma2 = Arma("Arco", "Ataque a distancia", 8)
        personaje1.establecerArma(arma1)
        personaje2.establecerArma(arma2)

        arma3 = arma1.clonar()
        personaje3 = personaje1.clonarSup()
        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")

        personaje3 = personaje1.clonarProf()

        if personaje1.esIgualSup(personaje3):
            print("Los personajes 1 y 3 son iguales (superficial).")
        else:
            print("Los personajes 1 y 3 son distintos (superficial).")
        if personaje1.esIgualProf(personaje3):
            print("Los personajes 1 y 3 son iguales (eval. profunda).")
        else:
            print("Los personajes 1 y 3 son distintos (eval. profunda).")
```

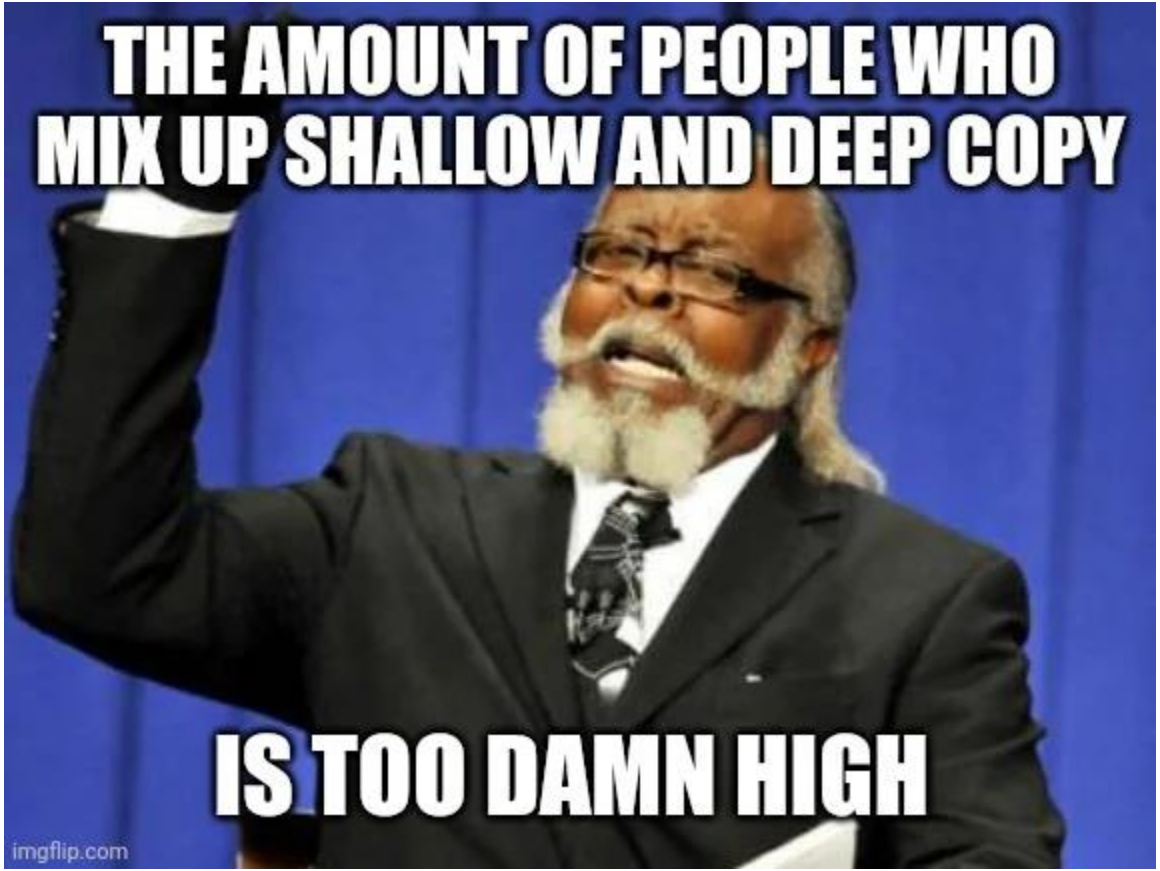


salida:

Los personajes 1 y 3 son iguales (superficial).

Los personajes 1 y 3 son distintos (superficial).

Los personajes 1 y 3 son iguales (eval. profunda).


THE AMOUNT OF PEOPLE WHO
MIX UP SHALLOW AND DEEP COPY
IS TOO DAMN HIGH
imgflip.com