***Explicacion clases***

Para comenzar, algo que tenemos que tener en cuenta es que, una clase por ser **abstracta**, no significa que **no puede tener un** initialize. Lo que nos diferencian las clases abstractas es que **no las instanciamos**, pero bien pueden tener un método initialize como cualquier otra (que usarán sus clases hijas).

Repasemos de paso, para qué nos sirve el método initialize; este nos permite que **apenas al instanciar un objeto** con la clase, sus atributos comiencen con un valor. Por ejemplo si tenemos:

class Persona

def initialize

@edad = 24

end

def edad

@edad

end

end

Tenemos el initialize que nos pone la edad de la persona con un valor de 24. Si ahora instanciamos una *persona*:

juani = Persona.new

El solo hecho de hacer Persona.new, nos inicializó la edad de *juani* a 24, por lo que si le preguntamos la edad:

juani.edad

=> 24

Nos devuelve 24.

Ahora, si queremos que sea una clase abstracta, y por ejemplo tenemos una nueva clase Estudiante que hereda de Persona:

class Estudiante < Persona

...

end

Como tiene **todos los métodos de la superclase** (porque la está heredando), **también tiene el** initialize. Esto significa que si hacemos:

pepita = Estudiante.new

¡También inicializó la edad en 24!

pepita.edad

=> 24

La clase abstracta nos sirve para justamente *abstraer* ciertos métodos, de forma que no tengamos por ejemplo que repetirlo en muchas clases diferentes, el initialize funciona de la misma forma, si queremos instanciar todas las clases hijas con el mismo valor inicial ¡podemos aprovechar la superclase! Así nos ahorramos tener exactamente el mismo método. Quizás lo podemos ver mejor con un ejemplo, si tenemos otra clase que herede de Persona:

class Profe < Persona

...

end

En lugar de hacer:

class Estudiante

def initialize

@eda = 24

end

end

class Profe

def initialize

@edad = 24

end

end

Implementamos la clase abstracta y nos quedamos con la versión que vimos anteriormente, donde ni Estudiante ni Profe tienen el initialize.

Con todo esto dicho, ¿qué pasa si ambas clases hijas comienzan con distinta edad inicial? Ya no podemos poner en un solo lugar el initializa porque ya **dejan de compartir el comportamiento**, ya no queremos que Estudiante y Profe hagan lo mismo, entonces lo tenemos que poner en cada una:

class Estudiante < Persona

def initialize

@eda = 24

end

end

class Profe < Persona

def initialize

@edad = 30

end

...

end

Si no vamos a instanciar nunca la clase Persona, y todas las clases hijas tienen un initialize (que recordemos que nos sirve para instanciar), ¿tiene sentido tener el initialize?

class Persona

def initialize

@edad = 24

end

def edad

@edad

end

end

¿En algún momento vamos a usarlo? ¡No! Entonces no necesitamos que exista en la clase abstracta:

class Persona

def edad

@edad

end

end

Y bueno... Si la edad cambia según la clase, ¿no podemos parametrizarla? Veamos el ejemplo con Estudiante

class Estudiante

def initialize(nueva\_edad)

@edad = nueva\_edad

end

end

Veamos como se instanciaría un Estudiante:

diana = Estudiante.new(24)

¿Y ese 24? ¡Es el parámetro! Cuando usamos parámetros, lo que estamos haciendo es obligar a que al momento se instanciar el objeto, tengamos que darle el valor. Esto significa que **cada vez que tenemos que instanciar un estudiante nuevo**, vamos a tener que pasarle el número 24 🧐. Y también podríamos pasarle cualquier otro número y no rompería nada:

diana = Estudiante.new(999)

Vemos que esto **no nos sirve** si queremos que **siempre se instancien con el mismo valor**. Si dejamos como parámetro la edad, perdemos el control, la persona usando el código puede poner lo que quiera. Y como vimos antes, queríamos que siempre se inicializara con 24, **queremos tener el control sobre eso**. A veces está bueno tener parámetros, en el caso de que queramos un valor **constante en todas las instancias**, no podemos hacerlo con parámetros, nos conviene implementar una solución como la que teníamos antes, en donde tanto el Estudiante como el Profe definan sus propios métodos initialize.

Disculpas por lo extenso de la respuesta, espero que te sirve y en todo caso, lo vemos por acá en más detalle.