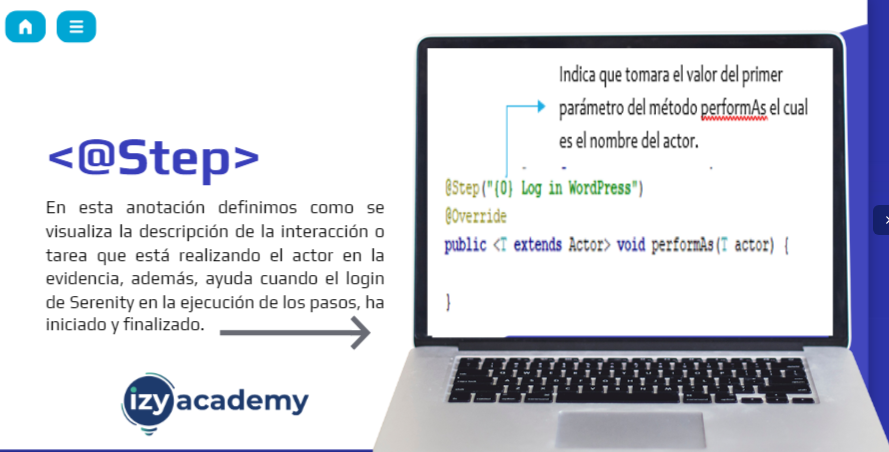
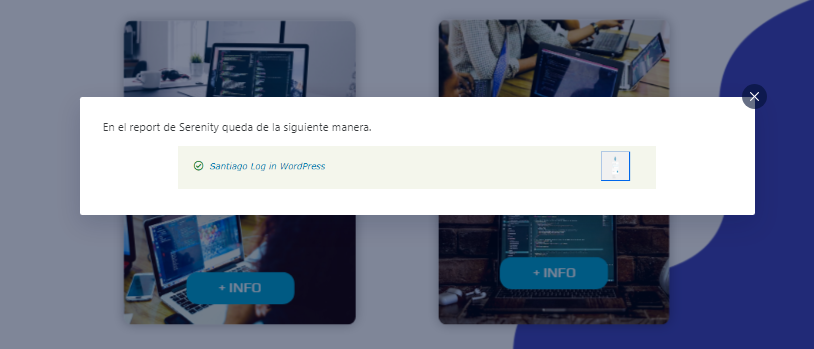
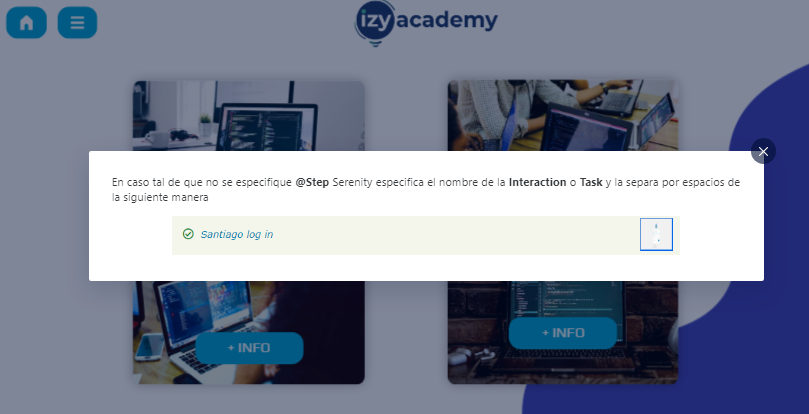
Anotaciones –

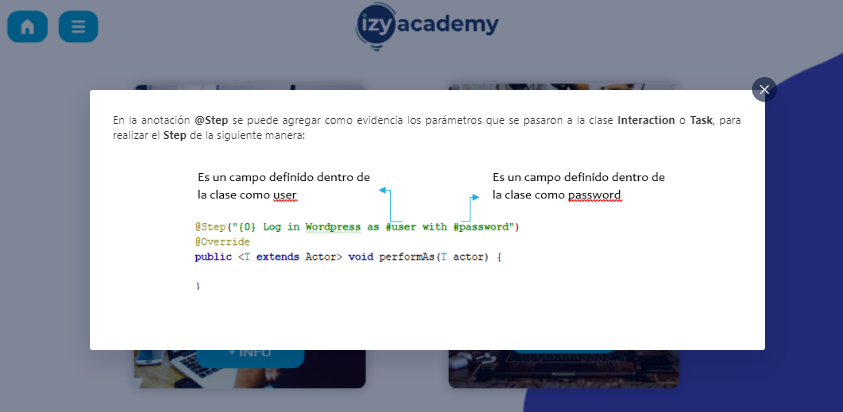




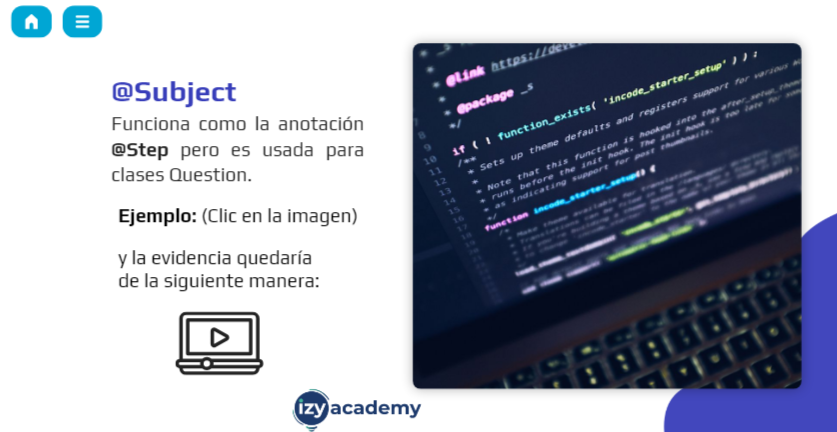




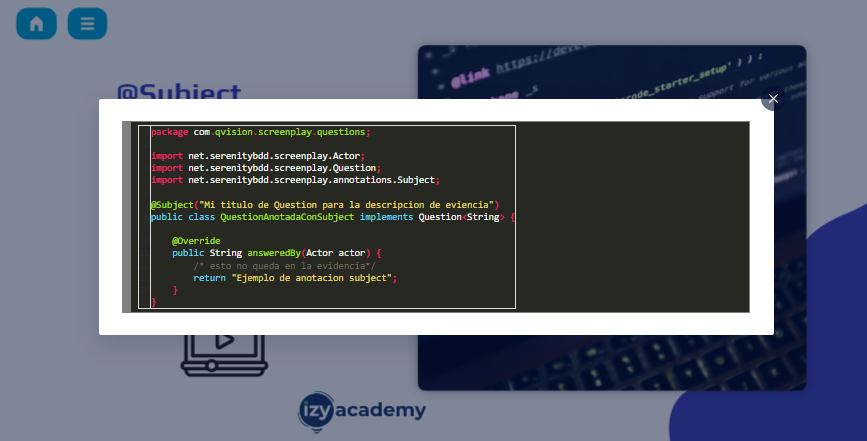


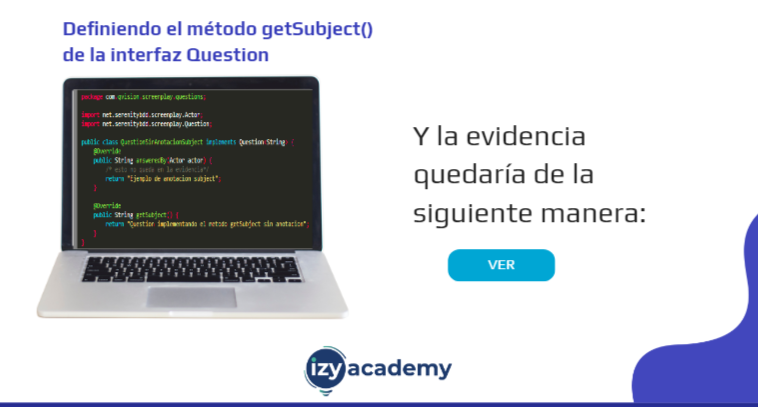


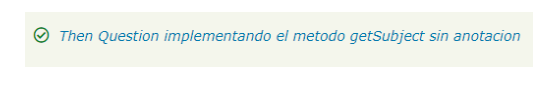














* Como se crea un actor -

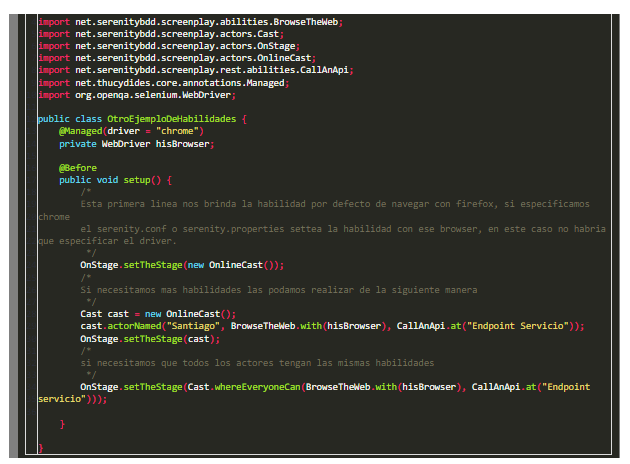




Como otorgar habilidades a un actor

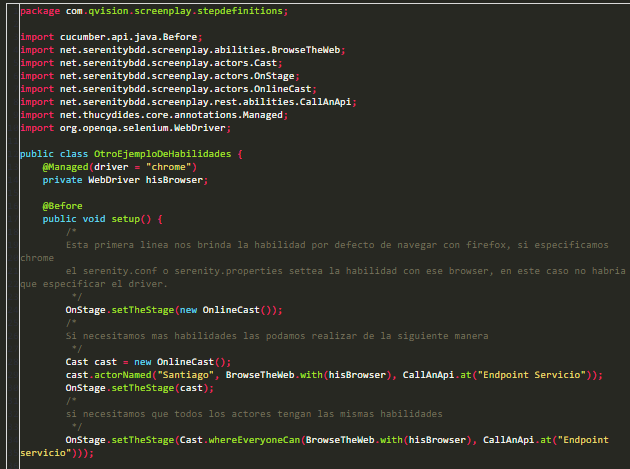


Como otorgar una habilidad a un actor



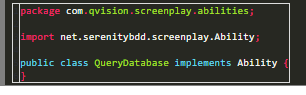
En este ejemplo indicamos que tipo de navegador vamos a utilizar, crear un atributo el cual se asignara a la instancia del navegador, ahora creamos al actor el cual se llamara “Santiago”

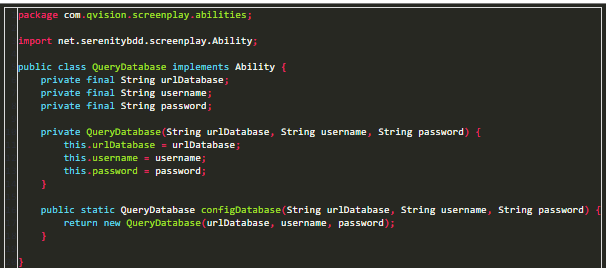
Y le indicamos que ahora tiene la habilidad de navegar en el browser de Chrome, todo esto antes de que se inicie el escenario



podemos crear nuevos elencos para nuestras pruebas y además podemos le agregamos al actor Santiago que posee dos nuevas habilidades las cuales son abrir el navegador que se encuentra almacenado en la variable de hisBrowser y realizar llamadas a un api con un endpPoint especificado

* CREAR HABILIDADES PARA UN ACTOR – documentación en el Blog de notas



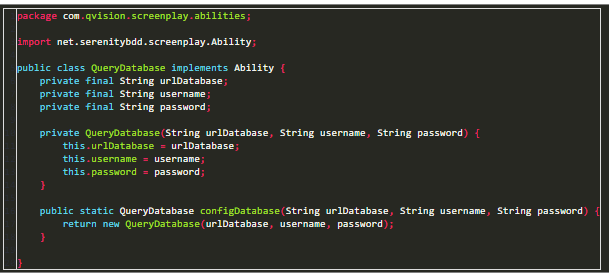


Hacemos uso de method factory el cual es un método que nos permite crear instancias de una clase sin necesidad del constructor además de que son maleables

* METHOD FACTORY

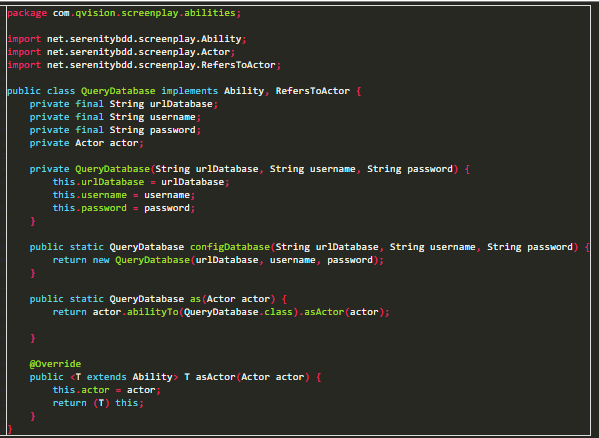
Este patrón creacional nos indica que siguiendo el principio solid de responsabilidad única

debemos crear todo lo referente a esta clase de manera privada y sus métodos deben ser estáticos

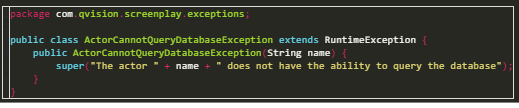


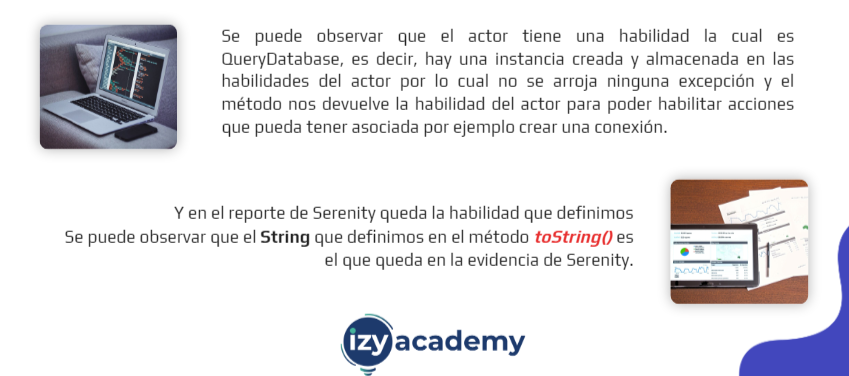
* COMO VERIFICAR SI UN ACTOR POSEE UNA HABILIDAD O NO

Con el método <ReferencesToActor> verificamos si una instancia de la habilidad creada está asociada al actor y si no debe arrojarnos una excepción



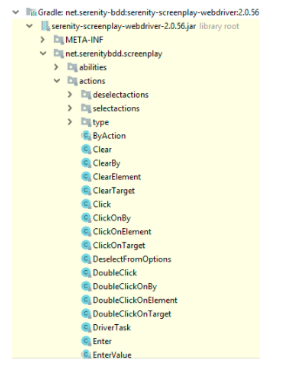
Además, para el manejo de esas excepciones debemos crear una clase la cual esta alojada en el paquete exceptions para poder manejar esas excepciones





* Como crear interacciones

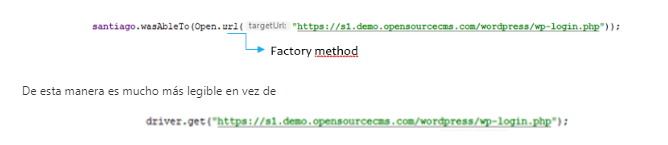
Para crear interacciones debemos haber implementado la interfaz interactions que a su vez hereda las interfaces action y performable, al indicar que el actor realiza una tarea o una acción debemos pasarle un objeto de tipo performable y ahora si indicamos que tipo de interacción vamos a realizar :



Para poder realizar interacciones en el test tras utilizar los métodos para indicar que son acciones debemos pasar un objeto de tipo performable el cual es como le dijéramos al programa que a un bloque amarrillo al pasarle un objetivo de tipo performable sea más rápido o en también si a un bloque azul al pasarle un objeto de tipo performable logramos que se vuelva mas lento

Las acciones deben tener el objeto performable para indicar que hacen algo

* De la siguiente manera podemos abrir el navegador con el patrón screenplay



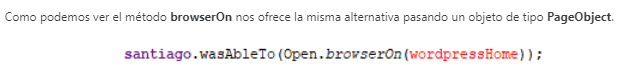
OJITO - Todas las interacciones son Factory methods

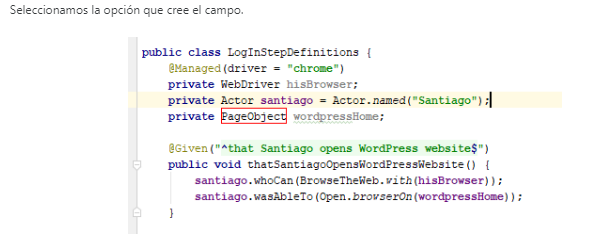
Recuerda – un factory method es una la implementación de un método de habilidad, esto con el fin de poder realizar una interaccion o una tarea

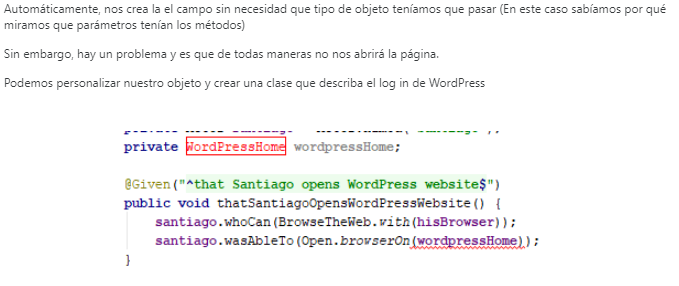
Cuando creamos interacciones necesitamos que herede performable para indicar que realizan algo

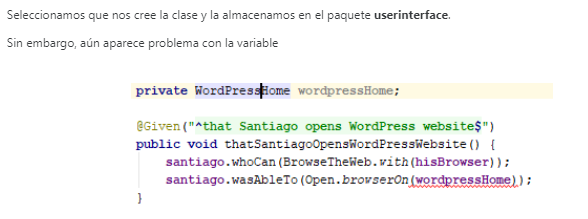
Otras formas de interacciones

También cuando queremos realizar interacciones podemos pasarle como argumentos page object como en el patron POM

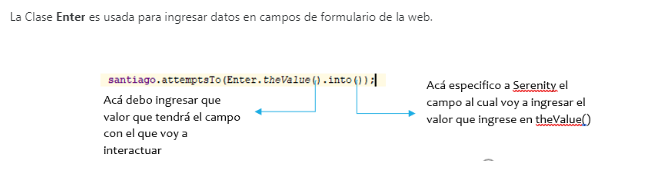


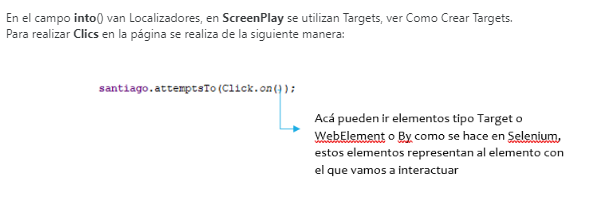


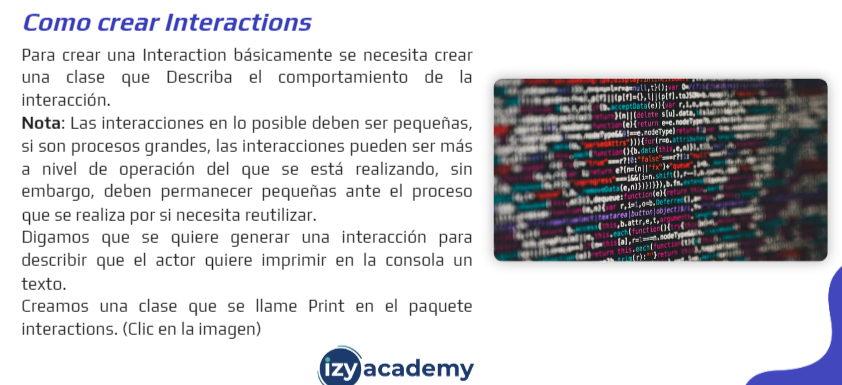


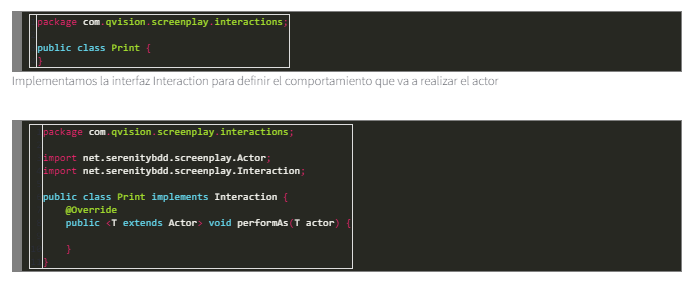


* Interacción con los Targets



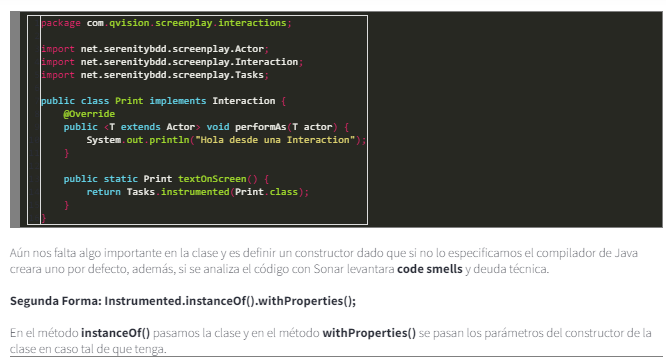




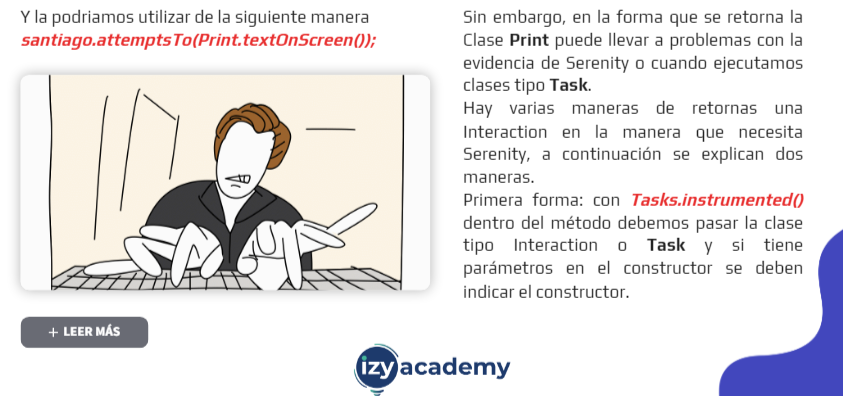


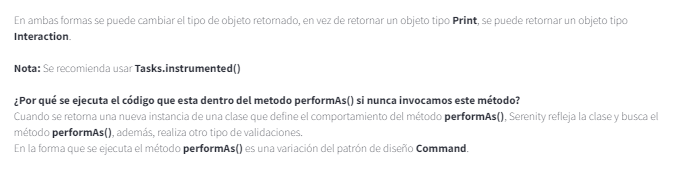
Creando una habilidad





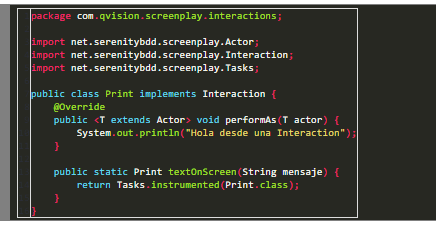






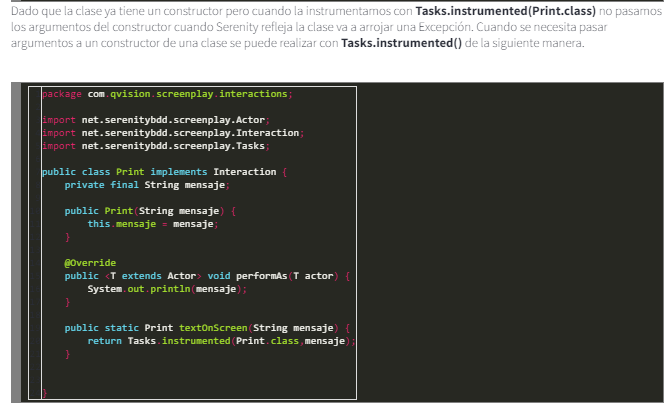


Clase estatica



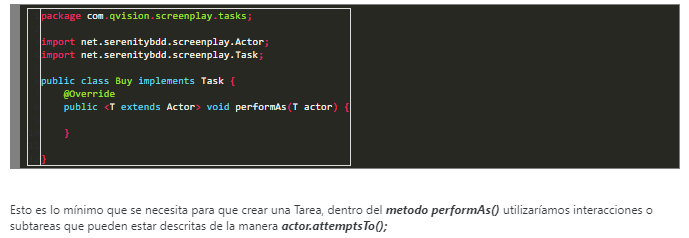


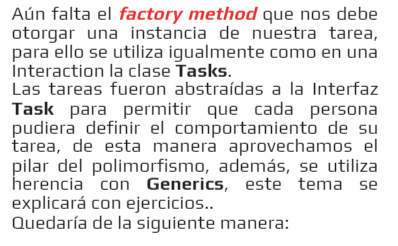




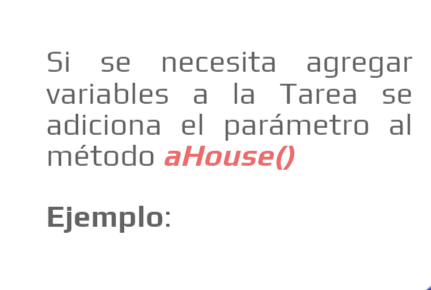


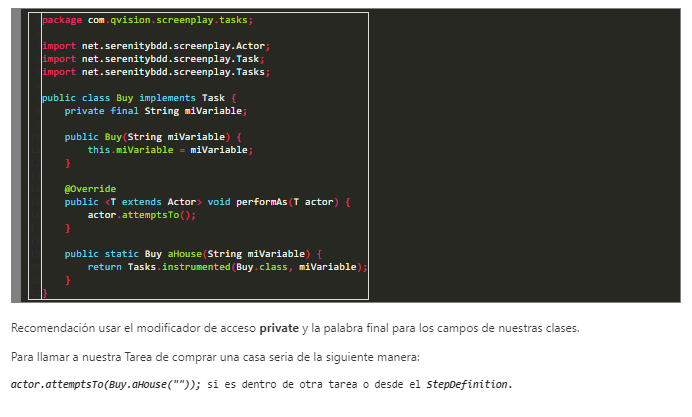
Crear tareas o tasks









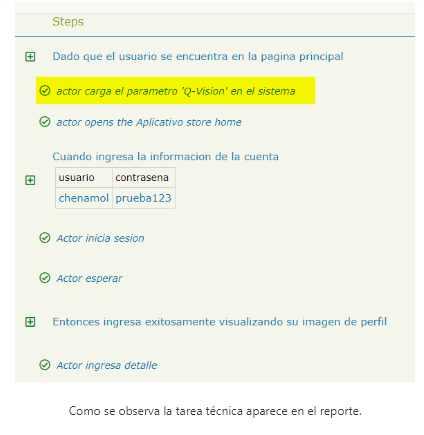


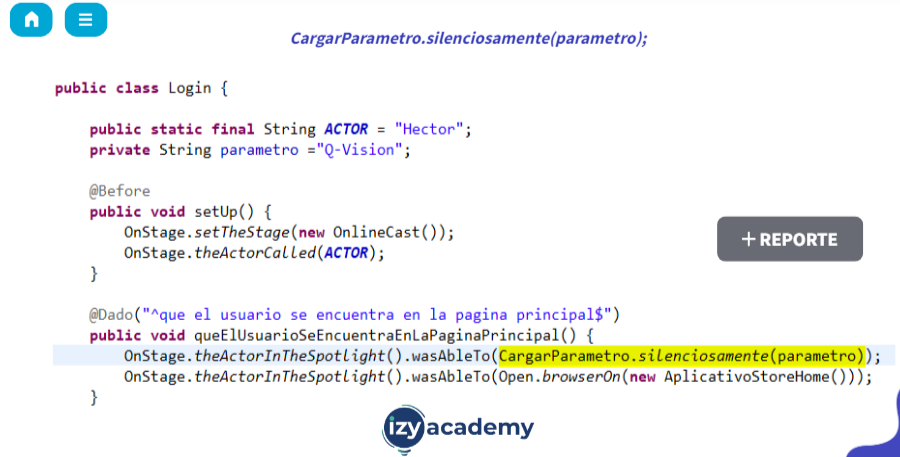
* Tareas Silenciosas

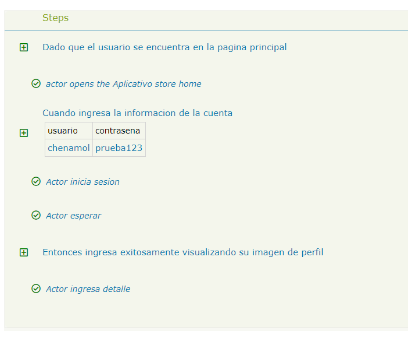


Literalmente es como si realizáramos tareas, pero no queremos que se muestren en el informe de serenity, determinamos si aparecen o no con el atributo isSilent () que es de tipo boolean, si es true no muestra la tarea en el informe, pero si es false si lo muestra

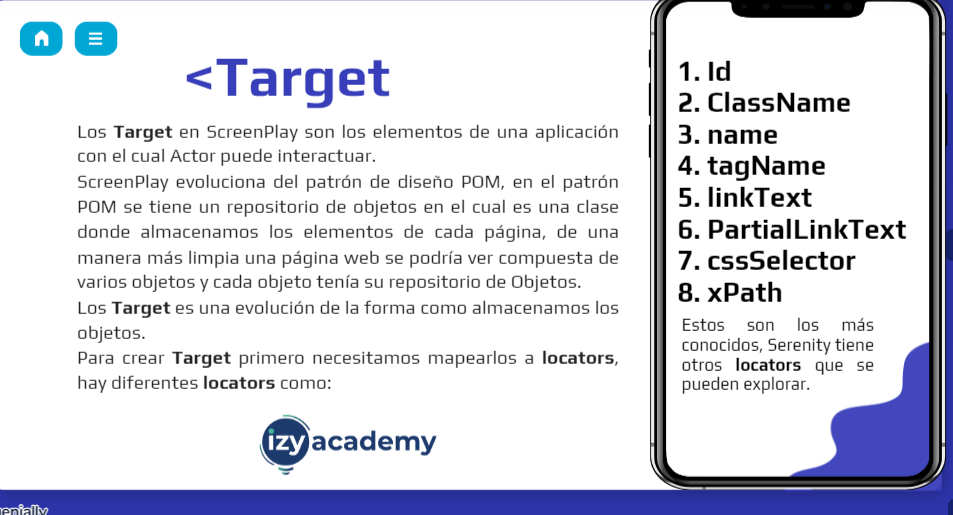
Asi aparece en los reportes

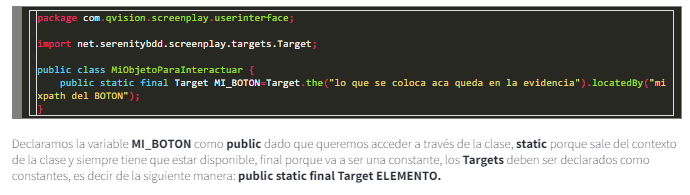




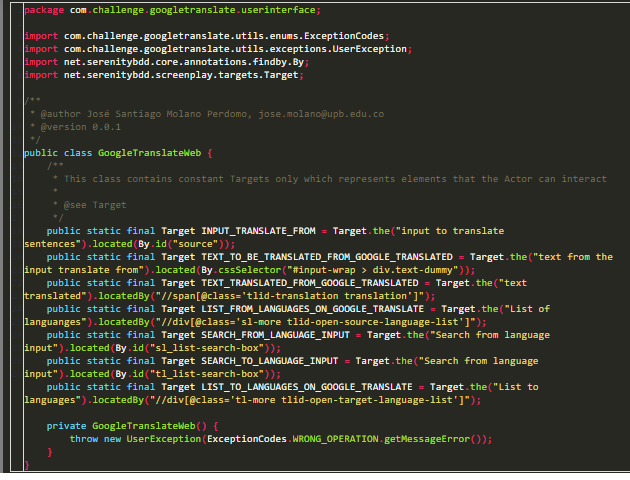


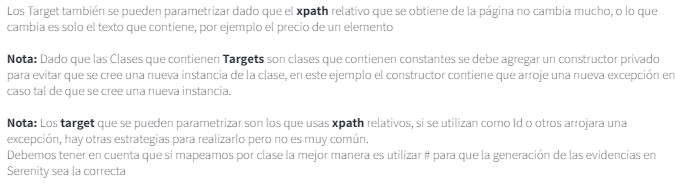
* Target

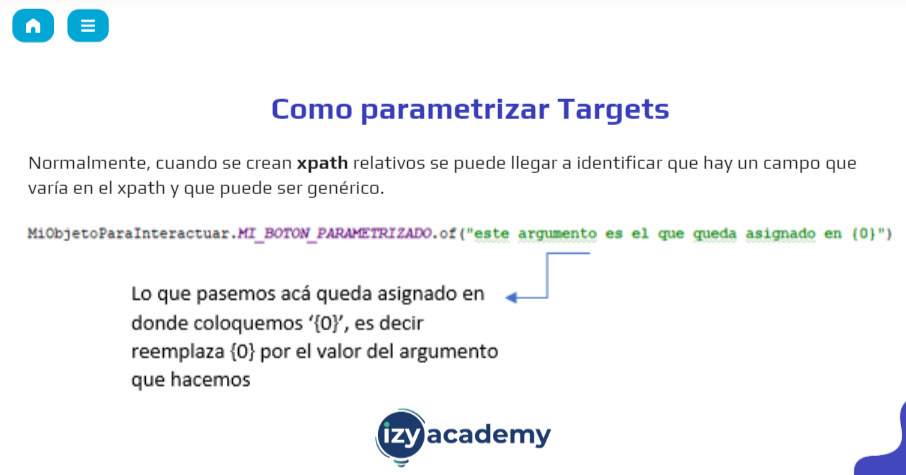


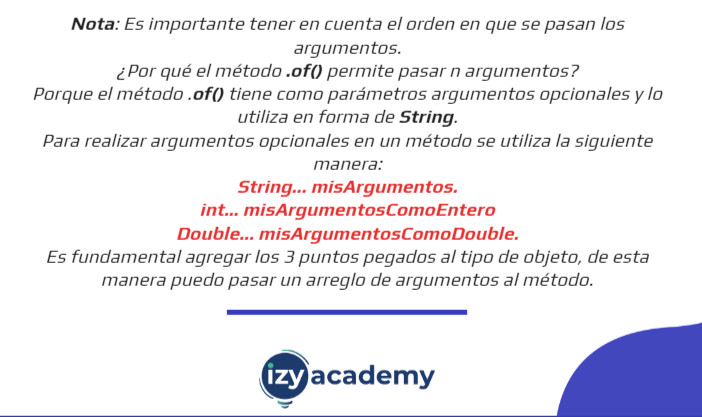


Creando varios tipos de target





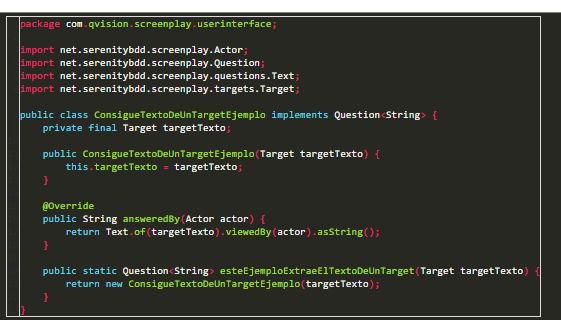


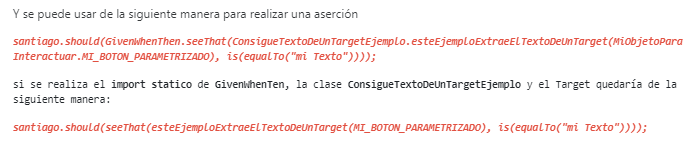
 

Tambien al of de la clase abstracta Target podemos pasarle arreglos de argumentos

* Questions

1 - forma





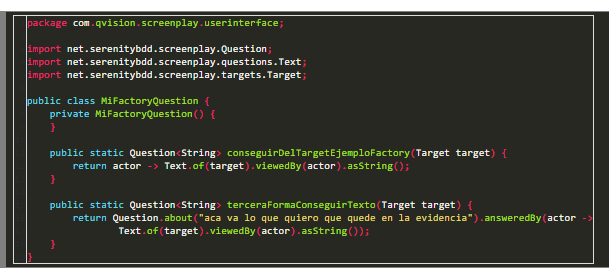
Como primera medida pedimos el objeto de tipo target, luego sacamos el contenido que este tenga un forma de string y luego verificamos si los textos son iguales

2 – forma





3 - forma





Con el primer método le indicamos que nos pase un objeto de tipo Target del cual vamos a sacar el texto que este contiene

PLUGGINS

