INGENIERIA INFORMATICA
Escuela Politécnica Superior
Universidad Autónoma De Madrid

Análisis y diseño de Software

Práctica 5

David Teófilo Garitagoitia Romero & Javier Fernandez de Alarcon Gervas

5/6/2021

Tabla de contenido

Ejercicio 1

Para este ejercicio el primer paso es la creación de la clase field la cual será genérica recibiendo el tipo genérico entre los símbolos T, en ella meteremos la variable f la cual será una función que permitirá el paso de un string a un objeto T, para que sea más general, haremos function<? Super String, ? extends T> es decir, será una función que permitirá pasar de un string o padre de string a un objeto T o subtipo de T, que es por tanto compatible con T.

Lo siguiente será juntar las validaciones con las excepciones, para facilitar la tarea creamos una clase Pair<A, B> que nos permite hacer parejas de elementos A y B, de esta forma, podremos hacer una lista de parejas predicate<T> (la misma T que se le pasa a Field ya que las comprobaciones se realizan sobre el mismo tipo de objeto del campo) y excepción, de esta forma tenemos cada predicado asociado con la excepción a lanzar, al llamar al método addValidation creamos una nueva pareja en la lista con el predicado y mensaje de la excepción.

A continuación, creamos el método SetValue cuyo objetivo es comprobar que el valor a establecer en ese campo cumple todas las condiciones, de no ser así lanzará la excepción pertinente y en caso contrario retornará un objeto del tipo tras realizar la función para cambiar de string (input del usuario) al objeto en cuestión y una función para crear respuestas a ese campo.

Por último, creamos el método equals para poder hacer comprobaciones con el contains y evitar repetidos.

Finalizada la clase Field pasamos a desarrollar el formulario, para el cual nos apoyaremos en dos clases nuevas, three similar a pair pero con 3 elementos para poder guardar la pregunta, el campo de pregunta y la respuesta del usuario, y, la clase respuesta que también será general con el genérico T que debe coincidir con el de su campo, esto es así ya que las respuestas se crean desde el propio Field y se añade al three dentro del exec siendo primeramente null al añadir la pregunta al no haberse introducido una respuesta todavía.

De esta forma el cuestionario estará formado por una lista de three formados por un string (la pregunta), un Field<?>, (? Ya que cada field es de un tipo diferente desconocido) y Respuesta<?>.

Además habrá un mapa que una las preguntas con las respuestas y que será el retorno de exec() método que se encargará de leer las respuestas del usuario por teclado, comprobar si son válidas, en cuyo caso, se genera la respuesta y se establece en ella el objeto casteado al tipo de respuesta retornado por SetValue que como ya hemos visto es un objeto del tipo genérico de la clase campo (el string tras ser convertido por la función) lo establece en la lista y el mapa, en caso contrario volverá a preguntar. Si el setValue se realiza si soltar una excepción significa que el valor cumple con todas las validaciones por lo que se pondrá succes a true para poder salir del bucle y pasar a la próxima pregunta, tras terminar con las preguntas retorna el mapa generado con el string de la pregunta y la respuesta del usuario.

Ejercicio 2

En este ejercicio se solicita la creación del DataAggregator y se especifica que guardará en orden, es por ello por lo que todos los tipos de campo (y por ende de respuesta) deben implementar compareTo para de esta forma facilitar la labor de ordenar datos, esto obliga a la clase address a implementarlo para así poder crear un field de tipo address.

El DataAgregator simplemente tendrá un mapa que asigne a cada pregunta la lista de respuestas del usuario que como implementan compareTo podrá ser ordenadas con el método Collections.sort();

Ejercicio 3

Para este ejercicio simplemente crearemos la clase abstracta AbstractForm que será padre de todos los formularios, e implementaremos un Proxy mediante la clase ProtectedForm que contendrá un formulario y dos strings, uno para guardar la contraseña y otro para guardar el último string introducido por el usuario además del número de intentos para poder introducrla, al crearlo simplemente se le pasa el formulario y la contraseña y se asignan los valores, el método protect llama al constructor y devuelve una instanciación de la clase, de esta forma al método exec() le podemos añadir la funcionalidad de la contraseña, primero comprueba si ya había introducido la contraseña correcta en cuyo caso ejecuta el método exec() de su formulario, en caso contrario solicita la contraseña hasta que introduzca la correcta o sobrepase los intentos permitidos, en caso de tener lugar la última, imprimirá el mensaje de error y retornará null en lugar del mapa esperado (podría también lanzarse una excepción pero esto implicaría modificar el main con un try catch o un throw Exception)

[FINAL DE DOCUMENTO]