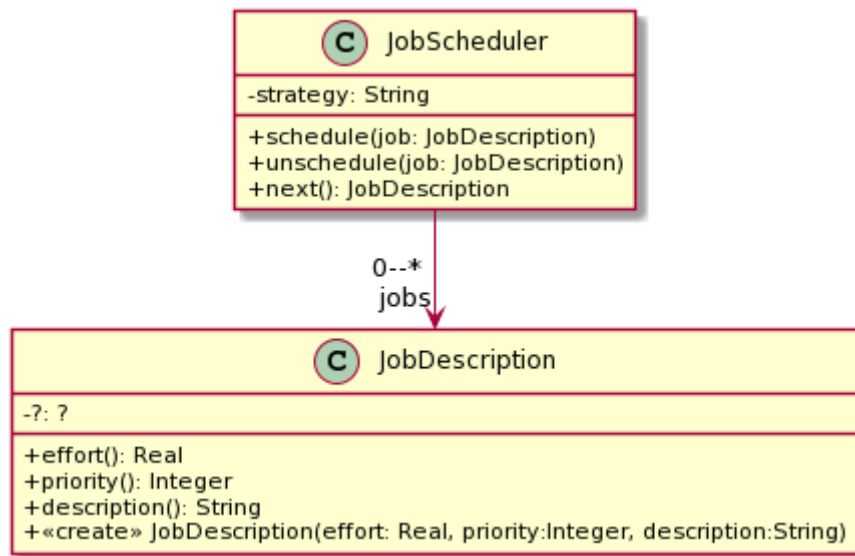


EJERCICIO 10 – JOB SCHEDULER

El JobScheduler es un objeto cuya responsabilidad es determinar qué trabajo debe resolverse a continuación. El siguiente diseño ayuda a entender cómo funciona la implementación actual del JobScheduler.



1. El mensaje `schedule(job: JobDescription)` recibe un job (trabajo) y lo agrega al final de la colección de trabajos pendientes.
2. El mensaje `next()` determina cuál es el siguiente trabajo de la colección que debe ser atendido, lo retorna, y lo quita de la colección.

En la implementación actual del método `next()`, el JobScheduler utiliza el valor de la variable `strategy` para determinar cómo elegir el siguiente trabajo. Dicha implementación presenta dos serios problemas de diseño:

- Secuencia de ifs (o sentencia switch/case) para implementar alternativas de un mismo comportamiento.
- Código duplicado.

a) Analice el código existente.

Utilice el [código y los tests](#) provistos por la cátedra y aplique lo aprendido (en particular en relación a herencia y polimorfismo) para eliminar los problemas mencionados. Siéntase libre de agregar nuevas clases como considere necesario. También puede cambiar la forma en la que los objetos se crean e inicializan. Asuma que una vez elegida una estrategia para un scheduler no puede cambiarse.

b) Verifique su solución con las pruebas automatizadas.

Sus cambios probablemente hagan que los tests dejen de funcionar. Corríjalos y mejórellos como sea necesario.