Saravanane MANICOME

Java Avancé

TP3 Interface fonctionnelle, la puissance du déclaratif.

Question 1

Pour renvoyer un résultat incertain, on utilise la classe Optional.

```
public class HealthCheck {
    @FunctionalInterface
    public interface URIFinder {
        Optional<URI> find();
    }
}
```

Question 2

fromArgument est une méthode public static de URIFinder.

```
static URIFinder fromArguments(String[] uris) {
    Objects.requireNonNull(uris);

    return () -> {
        if (uris.length == 0 || uris[0] == null)
            return Optional.empty();

        return URIfy(uris[0]);
    };
}
```

Question 3

```
static URIFinder fromURI(String uri) {
    Objects.requireNonNull(uri);
    return () -> URIfy(uri);
}

private static Optional<URI> URIfy(String uri) {
    try {
        return Optional.of(URI.create(uri));
    } catch (IllegalArgumentException e) {
        return Optional.empty();
    }
}
```

Saravanane MANICOME

Si une chaîne de caractères n’est pas valide ou que l’uri est mauvaise on retourne Optional.empty

Question 4

or est une méthode de URIFinder avec le modificateur default.

```
default URIFinder or(URIFinder other) {
    Objects.requireNonNull(other);

    return () -> {
        Optional<URI> optionalURI = this.find();
        if (optionalURI.isEmpty()) {
            optionalURI = other.find();
        }
        return optionalURI;
    };
}
```

Question 5

```
static URIFinder fromMapGetLike(String key, UnaryOperator<String> getUri) {
    Objects.requireNonNull(key);
    Objects.requireNonNull(getUri);

    return () -> {
        String uri = getUri.apply(key);

        if (uri == null)
            return Optional.empty();

        return URIfy(uri);
    };
}
```

Saravanane MANICOME

Question 6

```java
static <T> URIFinder fromMapGetLike(T key, Function<? super T, String> getUri) {
    Objects.requireNonNull(key);
    Objects.requireNonNull(getUri);

    return () -> {
        String uri = getUri.apply(key);

        if (uri == null)
            return Optional.empty();

        return URIfy(uri);
    };
}
```

Question 7

```java
static URIFinder fromPropertyFile(String key, Path path) {
    Objects.requireNonNull(key);
    Objects.requireNonNull(path);

    return () -> {
        Properties properties = new Properties();
        try (BufferedReader bufferedReader = Files.newBufferedReader(path)) {
            properties.load(bufferedReader);

            String property = (String) properties.get(key);
            if (property == null)
                return Optional.empty();

            try {
                return Optional.of(URI.create(property));
            } catch (IllegalArgumentException e) {
                return Optional.empty();
            }
        } catch (IOException e) {
            return Optional.empty();
        }
    };
}
```

Saravanane MANICOME

Question 8

```java
static boolean healthCheck(URI uri) throws InterruptedException {
    Objects.requireNonNull(uri);

    HttpClient httpClient = HttpClient.newBuilder().build();
    HttpRequest httpRequest = HttpRequest.newBuilder().uri(uri).build();
    try {
        HttpResponse<Void> httpResponse = httpClient.send(httpRequest, HttpResponse.BodyHandlers.discarding());
        return httpResponse.statusCode() == 200;
    } catch (IOException e) {
        return false;
    }
}
```