4 - Les suites de test

Plusieurs tests peuvent partager les même postulats initiaux

- un certain état en mémoire
- certaines variables définies
- une randomisation des états
  Les suites de tests permettent de ne pas repartir de zéro à chaque test.
  Une suite de test doit supporter l'itération automatisée de plusieurs scénarios.

Exemple

- pour chaque utilisateur:
- vérifier que son prénom est > 2 caractères
- que son nom est > 3 caractères
- que son nom complet est bien inféré et supérieur à 5 caractères

```
class Tests {
    constructor(testName) {
        this.testName = testName;
    }

    isLonger(expected, result) {
        console.log(`${this.testName}: ${result.length >
expected}`);
    }
};

class User {
    constructor(firstname, lastname) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.fullname = `${this.firstname} ${this.lastname}`;
    }
}

class UserTests {
    constructor(userObject) {
        this.user = userObject;
    }

    testFirstname() {
        new Tests("testFirstname")
            .isLonger(2, this.user.firstname);
    }

    testLastname() {
        new Tests("testLastname")
            .isLonger(3, this.user.lastname);
    }

    testFullname() {
        new Tests("testFullname")
            .isLonger(5, this.user.fullname);
    }

    runTests() {
        this.testFirstname();
        this.testLastname();
        this.testFullname();
    }
```

```
}

new UserTests(
    new User("Victor", "Hugo")
).runTests();

new UserTests(
    new User("Guy", "de Maupassant")
).runTests();

new UserTests(
    new User("JR", "Tolkien")
).runTests();
```

formaliser les tests d'un commerce ayant pour nécessité fonctionnelle:
avoir plus d'un rayon
avoir au moins un vendeur
avoir un nom qui soit composé de 5 lettres
en utilisant des classes pour faire matérialiser des Tests, une suite de test et un commerce.