# TDD avec dépendances



On s'intéresse au paiement des repas dans un restaurant d'entreprise. Le paiement se fait grâce à une caisse dans laquelle est insérée la carte personnelle de l'employé (aucune action n'est possible tant qu'aucune carte n'a été insérée).

La carte personnelle sert à la fois de porte-monnaie électronique et de support pour stocker les tickets fournis par l'employeur (pas nos tickets-repas nationaux, des tickets propres à l'entreprise). Tous les tickets d'une carte ont la même valeur, disons que la valeur est stockée sur la carte. Aucun découvert n'est possible. Un employé ne peut utiliser qu'un seul ticket lors de son passage en caisse.

La caisse permet de visualiser le solde du porte-monnaie électronique de la carte, ainsi que le nombre de tickets restants et leur valeur.

L'employé peut payer son repas de 2 manières :

**Sans utiliser de ticket** Le montant du repas est alors débité de sa carte.

**En utilisant un ticket** La carte est alors débitée du montant du repas moins la valeur d'un ticket, et un ticket est supprimé de la carte. La caisse ne rend pas d'argent, même si la valeur du ticket excède le montant du repas.

Répondez dans un README. Vosu pouvez réaliser le TP en Java, et quand il est fini expérimenter en Python, ou l'inverse.

**Q 1** : Comment allez-vous aborder le TP si vous pratiquez le TDD « à la mockist », le TDD « classique » (au sens de Fowler) ? □

**Q 2** : Listez les fonctionnalités de la classe `Caisse`, et indiquez pour chaque fonctionnalité ses différents comportements. □

**Q 3** : Vous êtes un mockist : réalisez en TDD la classe `Caisse`. □

**Q 4** : Réalisez en TDD la classe `Carte` en vous basant sur les méthodes prévues à la question précédente. □

**Q 5** : Réaliser quelques tests d'intégration. □