* 1. L’intention du patron template method est de définir un squelette d’un algorithme et laisser les sous-classes définir l’implémentation. Ce principe est appliqué lorsque la situation suivante est rencontrée : 2 classes possèdent plusieurs similitudes sans toutefois avoir la même implémentation pour toutes les méthodes. Le patron template method est parfait dans ce cas, permettant à chacune des classes d’avoir leur propre implémentation d’une méthode lorsque celles-ci diffèrent d’une à l’autre, tout en réunissant les 2 composants à l’aide d’une classe abstraite. Cette classe abstraite évite d’avoir d’implémenter les méthodes en double si celles-ci sont les mêmes pour les 2 sous-classes.

1. Il y a un avantage à définir la méthode *nettoyer*() comme méthode pure virtuelle et la méthode *removeTartre*() comme une méthode séparée dans l’implémentation de la classe *ElmCircuitLiquide*. En effet, pour toutes les classes héritant de *ElmCircuitLiquide,* l’implémentation de *removeTartre*() est la même. Toutefois, l’implémentation de la méthode *nettoyer*() n’est pas la même pour toutes les classes héritant de *ElmCircuitLiquide*. En effet, l’implémentation de *nettoyer*() est différente pour la classe *CircuitLiqComposite* qui, dans sa méthode *nettoyer*(), appelle la méthode *nettoyer*() sur chacun des objets *ElmCircuitLiquide* de son attribut vecteur. Ainsi, la méthode *nettoyer*() doit être pure virtuelle pour permettre une implémentation différente selon la classe qui hérite.
2. La méthode MachineAbs ::infuserThe() est un parfait exemple du patron de conception Template Method. En effet, cette méthode est pure virtuelle, permettant ainsi une implémentation différente pour les classes héritant, soit : *MachineBase* et *MachineLuxe*. Cette structure suit le principe du patron Template Method car on définit un squelette d’un algorithme et on laisse les sous-classes définir certaines étapes.