

TP 02 : Rapport Carte de crédit

Pour réaliser ce TP j'ai créé la classe **CarteCredit**.

CarteCredit contient les méthodes :

- doPost
- verifIP
- verifNum
- verifType
- verifDate
- readFile
- writeFile

CarteCredit:

doPost (public)

Entrée: HttpServletRequest requete, HttpServletResponse reponse

Sortie: void

Cette méthode est exécutée lors de l'appel de la classe **CarteCredit** elle charge les différents paramètres obtenus sur la page CarteCredit.html et appelle les méthodes suivantes les unes après les autres pour traiter les informations de manière séquentielle. D'abord verifDate, puis s'il n'y a pas d'erreur verifType, et s'il n'y a toujours pas d'erreur, verifNum.

Une fois cela achevé elle prend le verrou pour faire les opérations suivantes, qui vont lire et écrire dans le fichier transactions.txt se trouvant dans le répertoire WEB-INF (le serveur sur lequel se trouve cette servlet doit être un PC sinon il faut changer le signe du chemin d'accès). Si une erreur est présente dans une de ces vérifications la méthode verifIP est appelée, et si le String en retour n'est pas nul on l'enregistre dans transaction.txt.

Puis la transaction courante est enregistrée sur une nouvelle ligne du fichier transactions.txt

Le verrou est relâché.

Selon la valeur de l'erreur le String sortie est défini grâce à un switch, et enfin on décrit ce que la page à afficher va contenir. La variable sortie définit donc dynamiquement le résultat.

verifIP (private)

Entrée: String adresse, String transactions

Sortie: String

Cette méthode sert lorsqu'une erreur est détectée à vérifier si l'adresse de la machine distante s'est déjà trompée auparavant. Trois tentatives infructueuses amènent à être listé dans la première ligne du fichier transactions.txt. Pour contrôler si l'adresse (adresse) de la machine est déjà présente et s'est déjà fourvoyée j'ai utilisé des patterns qui se chargent de vérifier le texte (transactions) ligne par ligne. Ensuite s'il y a une modification à faire on reconstruit le fichier en y incluant l'adresse, si elle n'y est pas déjà présente, à la première ligne, et retourne le String contenant le fichier.

verifNum (private)

Entrée: String numero

Sortie: int

Cette méthode sert à vérifier si le numéro de la carte (numero) est correct en passant par un algorithme de contrôle. Si le numéro ne comporte pas 16 chiffres la valeur 4 est retournée et on s'arrête là. Sinon on sépare le String en 16, et on convertit chaque occurrence en int qu'on place dans un tableau (chiffres), si une exception apparaît dans cette opération cela veut dire qu'un des chiffres comportait un autre type, on attrape l'exception et on retourne 4.

Si tout se passe bien on continue la procédure différente pour les chiffres pairs et impairs, on additionne le tout et si le résultat est divisible par 10 on retourne 0, sinon 3.

verifType (private)

Entrée: String num1, String type

Sortie: int

Cette méthode sert à vérifier si le type de la carte correspond au numéro de la carte.

Elle est courte car l'utilisation des Patterns la simplifie beaucoup. Si le type est visa on compare que le premier chiffre est 4, si c'est une mastercard si les deux premiers chiffres sont 51, 52, 53, 54 ou 55. Si cela correspond on retourne 0, sinon 2. S'il n'y a pas de type de carte on retourne 4.

verifDate (private)

Entrée: String mois, String annee

Sortie: int

Cette méthode vérifie que la date entrée en paramètres est postérieure ou égale à la date de l'ordinateur. Pour cela on fait appel à la classe **GregorianCalendar** qui gère cela pour nous. J'ai du user d'un subterfuge pour la faire marcher car il faut additionner 100 à l'année (annee) qui était sous forme de 2 chiffres (ex : 105 pour 2005) alors que la documentation la présentait comme étant sous forme de 4 chiffres (ex : 1900). J'espère que ce n'est pas uniquement sur ma machine que ça marche ainsi... quoi qu'il en soit si le contrôle est positif la méthode retourne 0, sinon 1.

readFile (private)

Entrée: String name

Sortie: int

Cette méthode a été reprise telle quelle de la classe **Fichier** fournie dans la cours de programmation objet de l'été 2005. Elle sert à ouvrir un fichier (name) et à le rendre sous forme d'un String.

writeFile (private)

Entrée: String name, String text

Sortie: void

Cette méthode a été reprise telle quelle de la classe **Fichier** fournie dans la cours de programmation objet de l'été 2005. Elle sert à enregistrer un String (text) dans un fichier (name).

Pour le bonus j'ai créé la classe **Statistiques**.

Statistiques:

Statistiques contient les méthodes:

- doGet
- miseEnPage
- readFile

doGet (public)

Entrée: HttpServletRequest requete, HttpServletResponse reponse

Sortie: void

Cette méthode affiche le servlet. Si le fichier est vide on affiche « fichier vide » sinon on appelle la méthode miseEnPage en lui passant en paramètre le String obtenu par la méthode readFile à qui on demande d'ouvrir le fichier transactions.txt.

miseEnPage (private)

Entrée: String transactions

Sortie: String

Cette méthode met dynamiquement en page les informations continues dans la variable transactions. Pour cela on utilise principalement des Patterns en combinaison avec d'autres algorithmes subtils ce qui a pour effet de retourner un String de structure différente selon les informations apparaissant dans le String d'entrée.