Travail pratique #3

Ce travail doit être fait individuellement.

Aucun retard permis

Notions mises en pratique : le respect d'un ensemble de spécifications et contraintes, les classes et les objets, les tableaux, et les exceptions.

1. Spécifications

Dans la partie 1, votre travail consiste à implémenter une classe `CarnetTelephones` servant à modéliser un carnet de téléphones pouvant conserver les numéros de téléphone de vos contacts personnels. Dans la partie 2, votre travail consiste à modifier votre code pour ajouter la levée d'exceptions dans la classe `Telephone`, et pour ajouter la gestion de ces exceptions dans votre classe `CarnetTelephones`.

Les **classes fournies** pour faire ce TP (qui doivent se trouver dans votre projet, dans le paquetage par défaut) sont :

Fichier	Classe
`Telephone.java`	Classe modélisant le téléphone d'un contact personnel (À NE PAS MODIFIER, SAUF pour faire les modifications expliquées dans la partie 2).
`TelephoneInvalideException.java`	Classe d'exception utilisée par la classe `Telephone` (À NE PAS MODIFIER). Utilisée dans la partie 2.
`ExempleTestsCarnetTelephones.java`	Exemples de tests pour tester votre classe `CarnetTelephones` (après avoir complété la partie 2). Notez que ces tests ne couvrent qu'un échantillon des cas à tester. Vous devrez faire vos propres tests pour vous assurer de bien couvrir tous les cas à tester.

1.1 PARTIE 1 - Implémentation de la classe CarnetTelephones

Dans la classe `CarnetTelephones`, les téléphones sont stockés dans un tableau d'objets de type `Telephone` (attribut d'instance `lesTelephones`). Au départ, le tableau est de longueur 2, mais chaque fois qu'il n'y a plus de place pour ajouter un nouveau téléphone, on double la longueur du tableau.

Lorsqu'on **ajoute** un téléphone au tableau `lesTelephones`, on doit l'ajouter **dans la première case `null`** trouvée à partir du début du tableau. En effet, une case qui contient la valeur `null` signifie que cette case est vide. La **suppression** d'un téléphone à la case `j` du tableau `lesTelephones` consiste à affecter la valeur `null` à cette case (ce qui signifie qu'on vide la case). IL EST IMPORTANTE DE RESPECTER CES CONSIGNES SOUS PEINE DE PERDRE PLUSIEURS POINTS DANS LES TESTS.

La classe `CarnetTelephones` contient aussi l'attribut d'instance `nbrTelephones` qui sert à conserver le nombre de téléphones dans le tableau `lesTelephones`. La valeur de cet attribut doit être ajustée chaque fois qu'on ajoute ou supprime des téléphones.

La classe `CarnetTelephones` doit être implémentée selon les spécifications suivantes :

1.1.1 Constante de classe publique

Nom de la constante	Type	Description
`LNG_INIT_TAB`	`int`	Sert à initialiser la longueur du tableau `lesTelephones` (voir attributs d'instance). Cette constante **DOIT ÊTRE initialisée à 2**.

***NOTE** : Vous pouvez ajouter d'autres constantes de classe (publiques ou privées) si vous le jugez pertinent.*

1.1.2 Attributs d'instance

Nom de l'attribut	Type	Description
`lesTelephones`	`Telephone[]`	La liste des téléphones dans ce carnet. La longueur initiale de ce tableau doit être initialisée à 2, avec la constante de classe `LNG_INIT_TAB`. Au départ, lorsqu'il n'y a aucun téléphone dans ce carnet, toutes les cases de ce tableau sont initialisées à `null`.
`nbrTelephones`	`int`	Le nombre de téléphones dans ce carnet.

NOTES :
Les noms et types de tous les attributs mentionnés dans les tableaux ci-dessus doivent être respectés à la lettre sinon vous risquez de perdre plusieurs points dans les tests.

Les seuls attributs d'instance permis sont ceux mentionnés dans le tableau ci-dessus.

ATTENTION ! Les paramètres dans l'entête des méthodes suivantes doivent respecter l'ordre d'énumération dans les tableaux qui les décrivent. Le nom des méthodes ainsi que leur type de retour doivent aussi être respectés à la lettre. Sinon, vous risquez de perdre plusieurs points dans les tests.

1.1.3 Constructeur

Votre classe doit fournir un seul constructeur, sans paramètre, qui initialise un carnet de téléphones vide (ne contenant aucun téléphone). Le tableau `lesTelephones` doit contenir `LNG_INIT_TAB` cases, chacune étant initialisée à `null`.

***Note** : le constructeur peut être le constructeur par défaut ou bien vous pouvez l'implémenter explicitement.*

1.1.4 Méthodes d'instance publiques

Nom méthode : `obtenirTailleCarnet`
Type retour : `int`
Paramètres : aucun

Cette méthode permet d'obtenir le nombre de téléphones dans ce carnet de téléphones.

Nom méthode : `ajouterTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`tel`	`Telephone`	Le téléphone à ajouter à ce `CarnetTelephones`.

Permet d'ajouter le `tel` donné à ce `CarnetTelephones`. Si `tel` est `null`, il n'est pas ajouté. La méthode retourne `true` si le téléphone a bien été ajouté, `false` sinon.

Lorsque `tel` n'est pas `null` (on veut donc l'ajouter), cette méthode DOIT ajouter le `tel` dans la première case vide (`null`) trouvée à partir du début du tableau `lesTelephones`. Par exemple, si `lesTelephondes` = `[null, null]`, `tel` sera ajouté à la case 0, comme ceci : `[`**`tel`**`, null]`. Si `lesTelephones` = `[tel1, null, tel2, null]`, `tel` sera ajouté à la case 1, comme ceci : `[tel1, `**`tel`**`, tel2, null]`.

De plus, lorsque `tel` n'est pas `null` (on veut donc l'ajouter), et qu'il n'y a plus de place dans le tableau `lesTelephones`, la méthode doit agrandir le tableau `lesTelephones` en DOUBLANT son nombre de cases, avant d'ajouter `tel` au tableau. Par exemple, supposons que `lesTelephones` = `[tel1, tel2, tel3, tel4]`. Avant d'ajouter `tel`, on double d'abord le tableau qui devient :

`lesTelephones` **=** `[tel1, tel2, tel3, tel4, null, null, null, null]`.

Ensuite, on ajoute `tel` dans la première case libre, et le tableau devient :

`lesTelephones` **=** `[tel1, tel2, tel3, tel4,` **`tel`**`, null, null, null]`.

N'oubliez pas d'ajuster l'attribut d'instance `nbrTelephones` s'il y a lieu.

Nom méthode : `supprimerTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`ieme`	`int`	Indique le téléphone à supprimer dans ce `CarnetTelephones`.

Cette méthode permet de supprimer le i^{ème} téléphone dans ce `CarnetTelephones`. Si `ieme` est invalide (il ne correspond à aucun téléphone non `null`, dans le tableau `lesTelephones`), aucun téléphone n'est supprimé. La méthode retourne `true` si le téléphone a bien été supprimé, `false` sinon.

Note sur le paramètre ieme de cette méthode (et toutes les méthodes suivantes qui ont ce paramètre).
Le i^{ème} téléphone correspond au i^{ème} téléphone non `null` dans le tableau `lesTelephones`. Ce paramètre est valide s'il est compris entre 1 et `nbrTelephones` inclusivement. Par exemple, soit le tableau suivant contenant 4 téléphones :

`lesTelephones = [tel1, tel2, null, null, tel3, null, tel4, null]`

Le **1er** téléphone est `tel1`, le **2^{ième}** téléphone est `tel2`, le **3^{ième}** téléphone est `tel3`, et le **4^{ième}** téléphone est `tel4`.

Note sur le mécanisme de suppression dans le tableau `lesTelephones` **:**
Supprimer le téléphone se trouvant à un indice `j` consiste tout simplement à affecter la valeur `null` à la case `lesTelephone[j]`.

N'oubliez pas d'ajuster l'attribut d'instance `nbrTelephones` s'il y a lieu.

Nom méthode : `modifierNomTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`ieme`	`int`	Indique le téléphone dont on veut modifier le nom, dans ce `CarnetTelephones`.
`nouveauNom`	`String`	Le nouveau nom à assigner au `ieme` téléphone.

Cette méthode permet de modifier le nom de famille du i^{ème} téléphone dans ce `CarnetTelephones` par le nouveau nom donné en paramètre. Si `ieme` n'est pas valide c.-à-d. qu'il n'est pas entre 1 et `nbrTelephones` inclusivement, la modification n'a pas lieu. La méthode retourne `true` si la modification est effectuée, `false` sinon. Ne vous préoccupez pas, ici, de la validité du paramètre `nouveauNom`, on le fera dans la partie 2.

Nom méthode : `modifierPrenomTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`ieme`	`int`	Indique le téléphone dont on veut modifier le prénom, dans ce `CarnetTelephones`.
`nouveauPrenom`	`String`	Le nouveau prénom à assigner au `ieme` téléphone.

Cette méthode permet de modifier le prénom du i^{ème} téléphone dans ce `CarnetTelephones` par le nouveau prénom donné en paramètre. Si `ieme` n'est pas valide c.-à-d. qu'il n'est pas entre 1 et `nbrTelephones` inclusivement, la modification n'a pas lieu. La méthode retourne `true` si la modification est effectuée, `false` sinon. Ne vous préoccupez pas, ici, de la validité du paramètre `nouveauPrenom`, on le fera dans la partie 2.

Nom méthode : `modifierTypeTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`ieme`	`int`	Indique le téléphone dont on veut modifier le type, dans ce `CarnetTelephones`.
`nouveauType`	`int`	Le nouveau type à assigner au `ieme` téléphone.

Cette méthode permet de modifier le type du i^{ème} téléphone dans ce `CarnetTelephones` par le nouveau type donné en paramètre. Si `ieme` n'est pas valide c.-à-d. qu'il n'est pas entre 1 et `nbrTelephones` inclusivement, la modification n'a pas lieu. La méthode retourne `true` si la modification a bien eu lieu, `false` sinon. Ne vous préoccupez pas, ici, de la validité du paramètre `nouveauType`, on le fera dans la partie 2.

Nom méthode : `modifierNoTelephone`
Type retour : `boolean`

Paramètre	Type	Description
`ieme`	`int`	Indique le téléphone dont on veut modifier le numéro, dans ce `CarnetTelephones`.
`nouveauNoTel`	`String`	Le nouveau numéro à assigner au `ieme` téléphone.

Cette méthode permet de modifier le numéro du i^{ème} téléphone dans ce `CarnetTelephones` par le nouveau numéro donné en paramètre. Si `ieme` n'est pas valide c.-à-d. qu'il n'est pas entre 1 et `nbrTelephones` inclusivement, la modification n'a pas lieu. La méthode retourne `true` si la modification a bien eu lieu, `false` sinon. Ne vous préoccupez pas, ici, de la validité du paramètre `nouveauNoTel`, on le fera dans la partie 2.

Nom méthode : `rechercherTelephones`
Type retour : `Telephone[]`

Paramètre	Type	Description
`motif`	`String`	Le motif à trouver dans le nom ou le prénom des téléphones à retourner.

Cette méthode permet de rechercher les téléphones dont le nom ou le prénom contient le `motif` donné. La recherche ne doit pas tenir compte de la casse. La méthode retourne un tableau contenant les téléphones trouvés. La longueur du tableau retourné doit être égale au nombre de téléphones trouvé : si aucun téléphone n'est trouvé, le tableau retourné est de longueur 0, si 1 téléphone est trouvé, le tableau retourné est de longueur 1, si 2 téléphones sont trouvés, le tableau retourné est de longueur 2, etc.

Si le motif donné en paramètre est `null` ou de longueur égale à 0, le tableau retourné est de longueur 0. De plus, les téléphones dans le tableau retourné doivent conserver l'ordre dans lequel ils apparaissent dans le tableau `lesTelephones`.

Par exemple, supposons un carnet qui contient les 5 téléphones suivants :

```
Lucie Bertrand : 234-7676 [DOMICILE]
luc bertrand : 123-7853 [AUTRE]
Mirabelle Bertrand : 654-9898 [CELLULAIRE]
Isabelle Miron : 298-3422 [TRAVAIL]
Jessie Coutu : 555-4444 [DOMICILE]
```

Voici le résultat de quelques recherches :

Appel	**Tableau retourné**

`rechercherTelephones("BERT")`	`[`
	`  Lucie **Bert**rand : 234-7676 [DOMICILE]`
	`  luc **bert**rand : 123-7853 [AUTRE]`
	`  Mirabelle **Bert**rand : 654-9898 [CELLULAIRE]`
	`]`

`rechercherTelephones("mir")`	`[`
	`  **Mir**abelle Bertrand : 654-9898 [CELLULAIRE]`
	`  Isabelle **Mir**on : 298-3422 [TRAVAIL]`
	`]`

`rechercherTelephones("belle")`	`[`
	`  Mira**belle** Bertrand : 654-9898 [CELLULAIRE]`
	`  Isa**belle** Miron : 298-3422 [TRAVAIL]`
	`]`

`rechercherTelephones("JEssiE")`	`[`**`Jessie`** `Coutu : 555-4444 [DOMICILE]]`
`rechercherTelephones("hugo")`	`[]  //tableau de longueur 0.`

Nom méthode : `obtenirTelephone`
Type retour : `Telephone`

Paramètre	**Type**	**Description**
`ieme`	`int`	Indique le téléphone de ce `CarnetTelephones` à retourner.

Cette méthode permet d'obtenir le i^{ème} téléphone dans ce `CarnetTelephones`. Si `ieme` n'est pas valide c.-à-d. qu'il n'est pas entre 1 et `nbrTelephones` inclusivement, la méthode retourne `null`.

Nom méthode : `viderCarnet`
Type retour : `void`
Paramètre : aucun

Cette méthode supprime (voir méthode `supprimerTelephone`) TOUS les téléphones de ce `CarnetTelephones`. Après l'appel de cette méthode, le carnet est vide, et la taille du carnet est 0 (la taille étant le nombre de téléphones dans le carnet).

***Note** : la longueur du tableau `lesTelephones` n'est pas modifiée.*

Nom méthode : `fusionnerCarnets`
Type retour : `void`

Paramètre	Type	Description
`autreCarnet`	`CarnetTelephones`	Le carnet à fusionner avec ce `CarnetTelephones`.

Cette méthode ajoute, dans ce carnet de téléphones, tous les téléphones (non `null`) contenus dans `autreCarnet`. Les téléphones doivent être ajoutés dans CE carnet dans l'ordre où ils apparaissent dans `autreCarnet` : du premier au dernier. L'ajout de chaque téléphone doit se faire comme spécifié dans la description de la méthode `ajouterTelephone`. Il se peut que la fusion crée des doublons dans ce `CarnetTelephones`, mais vous n'avez pas à vous en préoccuper.

Si `autreCarnet` est `null`, la méthode ne fait rien, et ce `CarnetTelephones` demeure inchangé.

Précisions sur l'implémentation de la classe `CarnetTelephones`

- Vous devez respecter le principe d'encapsulation des données.
- Les méthodes énumérées dans cette section sont les seules méthodes publiques qui doivent se trouver dans la classe `CarnetTelephones`. **Vous pouvez (et devriez) cependant ajouter vos propres méthodes privées pour bien découper votre code (séparation fonctionnelle).**
- Les seules variables globales permises sont les attributs d'instance énumérés à la section 1.1.2 et la constante de classe (voir section 1.1.1). Rappel : vous pouvez ajouter des constantes de classe publiques ou privées si vous le jugez pertinent.

1.2 PARTIE 2 - Levée et gestion d'exceptions

Dans cette partie, vous devez lever des exceptions dans la classe `Telephone`, et vous devez les gérer dans votre classe `CarnetTelephones`. Voici les modifications à apporter.

Modifications à faire dans la classe `Telephone` :

- Modifiez le constructeur pour qu'il lève une `TelephoneInvalideException` si au moins l'un de ses paramètres est invalide. Lisez la Javadoc du constructeur pour connaître les conditions de validité de chaque paramètre.
- Modifiez les méthodes `setNom`, `setPrenom`, `setNoTel`, et `setType` pour que chacune de ces méthodes n'effectue aucune modification, et lève plutôt une `TelephoneInvalideException` lorsque son paramètre est invalide. Lisez la Javadoc de la méthode pour connaître les conditions de validité de son paramètre.

ATTENTION ! Les seules modifications permises dans la classe `Telephone` sont celles mentionnées ci-dessus. Vous pouvez cependant ajouter des méthodes privées dans cette classe si vous le jugez pertinent (pour la séparation fonctionnelle).

N'oubliez pas de compléter la Javadoc des méthodes modifiées…

Modifications à faire dans la classe `CarnetTelephones` :

- Dans la methode `modifierNomTelephone`, vous devez valider le paramètre `nouveauNom`. Si le paramètre `nouveauNom` est invalide, la modification ne doit pas avoir lieu (et la méthode doit retourner `false` dans ce cas). Un `nouveauNom` valide est une chaine de caractères non `null` dont la longueur est strictement plus grande que 0. **Utilisez la gestion des exceptions (`try`… `catch`) pour déterminer la validité de `nouveauNom`.**
- Dans la methode `modifierPrenomTelephone`, vous devez valider le paramètre `nouveauPrenom`. Si le paramètre `nouveauPrenom` est invalide, la modification ne doit pas avoir lieu (et la méthode doit retourner `false` dans ce cas). Un `nouveauPrenom` valide est une chaine de caractères non `null` dont la longueur est strictement plus grande que 0. **Utilisez la gestion des exceptions (`try`… `catch`) pour déterminer la validité de `nouveauPrenom`.**
- Dans la methode `modifierTypeTelephone`, vous devez valider le paramètre `nouveauType`. Si le paramètre `nouveauType` est invalide, la modification ne doit pas avoir lieu (et la méthode doit retourner `false` dans ce cas). Un `nouveauType` valide est un entier entre 0 et `Telephone.TYPE_TEL.length - 1` inclusivement. **Utilisez la gestion des exceptions (`try`… `catch`) pour déterminer la validité de `nouveauType`.**
- Dans la methode `modifierNoTelephone`, vous devez valider le paramètre `nouveauNoTel`. Si le paramètre `nouveauNoTel` est invalide, la modification ne doit pas avoir lieu (et la méthode doit retourner `false` dans ce cas). Un `nouveauNoTel` valide est une chaine de caractères non `null` dont la longueur est 7, 10 ou 11. **Utilisez la gestion des exceptions (`try`… `catch`) pour déterminer la validité de `nouveauNoTel`.**

ATTENTION ! Vous NE DEVEZ PAS refaire le code des validations qui est déjà fait dans les *setters* de la classe `Telephone` pour valider le nom, le prénom, le type de téléphone, et le numéro de téléphone.

1.3 TESTS de la classe CarnetTelephones

Vous n'avez pas à concevoir d'application dans ce TP, cependant, vous devrez concevoir des tests pour vérifier vos méthodes. Pour ce faire, vous pouvez créer une autre classe, disons la classe `TestsCarnetTelephones`, dans laquelle vous aurez une méthode `main` exécutant les tests de chacune des méthodes publiques de votre classe `CarnetTelephones`. Vous devez tester vos méthodes avec tous les cas possibles de paramètre(s) (valeurs valides et invalides) et vérifier que vos méthodes réagissent correctement (comme spécifié). Pour construire vos propres tests, vous pouvez vous référer à la classe `ExempleTestsCarnetTelephones` fournie avec l'énoncé de ce TP. Notez que cette classe est un exemple, et ne teste pas tout. Votre classe de tests n'est pas à remettre.

2. Précisions supplémentaires

- Vous ne DEVEZ PAS modifier la classe `TelephoneInvalideException`.car vos classes seront testées avec cette classe, telle quelle.
- N'oubliez pas d'écrire les commentaires de documentation (Javadoc) pour chacune de vos méthodes (sauf la méthode `main`).
- Vous devez respecter le principe d'encapsulation.
- Vous DEVEZ respecter le style Java, et les bonnes pratiques de programmation.
- Vos classes doivent se trouver dans le paquetage par défaut.
- Il ne doit y avoir aucune saisie et aucun affichage dans les méthodes des classes à remettre.
- Prenez soin de bien découper le code des méthodes publiques à l'aide de méthodes **privées** cohésive, spécialisées, et bien paramétrées.
- Les seules CLASSES PERMISES sont `Telephone`, `CarnetTelephones`, `TelephoneInvalideException`, `String`, `Math`, `Character`, `Double`, et `Integer`. Vous pouvez évidemment aussi utiliser les tableaux.
- Utilisez des constantes (`final`) autant que possible
- Les constantes doivent être déclarées et initialisées au niveau de la classe (constantes globales) : `public static final`…
- Vous DEVEZ utiliser les notions vues au cours pour faire votre TP.

- Votre code doit compiler et s'exécuter avec le JDK 7. Tout ce qu'on voit au cours respecte cette consigne.
- Votre fichier de code source doit être encodé en UTF-8.
- **N'oubliez pas d'écrire (entre autres) votre nom complet et votre code permanent dans l'entête de votre classe.**

Le non-respect de toute spécification ou consigne se trouvant dans l'énoncé du TP (et les exemples d'exécution donnés avec l'énoncé du TP) est susceptible d'engendrer une perte de points.

Note : *Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.*

3. Détails sur la correction

3.1 La qualité du code (40 points)

Concernant les critères de correction du code, **lisez attentivement** le document "**CriteresGeneraux DeCorrection_v2.pdf**". De plus, votre code sera noté sur le respect des conventions de style Java vues en classe (dont un résumé se trouve dans le document "**ConventionsStyleJavaPourINF1120_INF2120.pdf**". Ces deux documents peuvent être téléchargés dans la section TRAVAUX PRATIQUES (ET BOITES DE REMISE) de la page Moodle du cours.

Votre code sera corrigé sur la totalité ou une partie des critères de correction indiqués ci-dessus, ainsi que sur le respect des spécifications/consignes mentionnées dans ce document.

3.2 L'exécution (60 points)

Un travail qui ne compile pas se verra attribuer la note 0 pour l'exécution.

Votre code sera testé en tout ou en partie. Les points seront calculés sur les tests effectués. Ceci signifie que si votre code fonctionne dans un cas x et que ce cas x n'est pas testé, vous n'obtiendrez pas de points pour ce cas. Il est donc dans votre intérêt de vous assurer du bon fonctionnement de votre programme dans tous les cas possible. Notez que les tests fournis ne couvrent pas tous les cas à tester.

4. Date et modalité de remise

4.1 Remise

Date de remise : au plus tard le 22 avril 2024 à 23h59 - **AUCUN RETARD PERMIS**.
Les deux fichiers à remettre :

- `Telephone.java` (après y avoir fait les modifications de la partie 2)
- `CarnetTelephones.java` (après avoir fait la partie 1 et la partie 2)
- NE PAS ARCHIVER (zip, rar, …) VOS FICHIERS à remettre (les remettre séparément).

Remise via Moodle uniquement.
Vous devez remettre (téléverser) vos fichiers sur le site du cours (Moodle). Vous trouverez la boîte de remise dans la section **TRAVAUX PRATIQUES (ET BOITES DE REMISE) – BOITE DE REMISE DU TP3**.

Vérifiez deux fois plutôt qu'une que vous avez remis les bons fichiers, car c'est ceux-là qui seront considérés pour la correction.

Assurez-vous de remettre votre travail avant la date limite, sur Moodle, car sinon vous n'aurez plus accès à la boite de remise, et vous ne pourrez plus remettre votre travail.

4.2 Politique concernant les retards

AUCUN RETARD NE SERA ACCEPTÉ.

Aucun travail ne sera accepté après la date limite de remise (22 avril 2024 à 23h59).

4.3 Remarques générales

- **Aucun programme reçu par courriel ne sera accepté**. Plus précisément, un travail reçu par courriel sera considéré comme un travail non remis.

- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Dropbox, Google Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir une extension pour la remise de votre travail.

- **N'oubliez pas d'écrire (entre autres) votre nom complet et votre code permanent dans l'entête des classes à remettre.**

N'attendez pas à la dernière minute pour commencer le travail, vous allez fort probablement rencontrer des problèmes inattendus!

Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !