CÉGEP STE-FOY – AUTOMNE 2018
PROGRAMMATION OBJET – 420-V29-SF

Énoncé du Travail Pratique 2

14 novembre 2018

Préparé par
Benjamin Lemelin

1 Résumé

Développer une application permettant de convertir un entier en chiffres romains.

2 Conditions de réalisation

Valeur de la note finale	Type	Durée	Nombre de remises
5 %	Individuel	7 jours	1

3 Spécifications

3.1 Projet de départ

Créez un nouveau projet Android sous Android Studio. Placez-le ensuite dans un nouveau dépôt Git et donnez-en accès à votre professeur le plus tôt possible.

3.2 Fonctionnalités

L'application doit permettre de convertir un nombre en chiffres romains. L'application se doit d'être robuste et donc refuser toute entrée invalide.

3.3 Tests Unitaires

Les tests unitaires de la couche modèle doivent être faits.

3.4 Tests Android

Vous n'avez pas à écrire de tests d'interface.

3.5 Interface utilisateur

La maquette est en français et vous devrez faire l'interface en français (obligatoire).

C'est ici que l'usager saisit le chiffre à convertir. Seuls les chiffres entre 1 et 4999 inclusivement devraient être acceptés par ce champ.

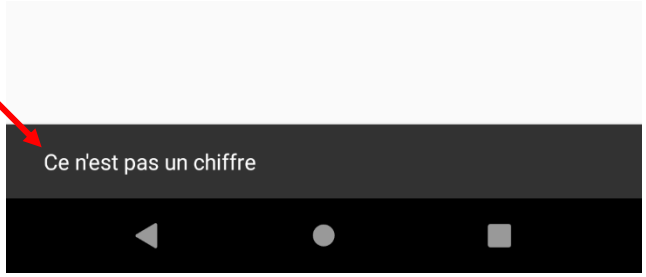
Ce bouton effectue la conversion du chiffre en chiffres romains.

L'interface au grand complet devrait utiliser un « ConstrainLayout ». En fait, il vous est même interdit d'utiliser quoi que ce soit d'autre.

Le texte ici est plus gros. Pour le rendre plus gros, changez la propriété « TextAppearance » pour « Display4 ».

N'oubliez pas de gérer les changements d'orientation de l'écran.

Si un erreur survient, utilisez un « Snackbar » pour afficher un message d'erreur. N'oubliez pas que les chaines de caractères doivent se trouver dans le fichier « strings.xml ».



3.6 Chiffres romains

Les chiffres romains utilisent les symboles suivants :

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Les symboles sont répartis en 2 catégories : les unités (**I**, **X**, **C** et **M**) et les demies (**V**, **L** et **D**). Les unités peuvent être répétées jusqu'à 3 fois pour créer un chiffre, tel que **III** pour 3 et pour **XX** 20.

Comme il n'est pas permis écrire **IIII** pour 4, il faut utiliser la demie la plus proche auquel on retire une unité. Par exemple, **IV** pour 4 et **XIX** pour 19. Pour écrire un nombre plus grand qu'une demie, il faut ajouter des unités (jusqu'à trois unités), tel que **VII** pour 7 et **LII** pour 52. Encore une fois, puisqu'il n'est pas permis d'écrire **VIIII** pour 9, il faut écrire **IX**.

Seule exception à la règle : la lettre **M** peut être répétée jusqu'à 4 fois. Sachant cela, la valeur maximale pouvant être représentée en chiffres romains est 4999, soit **MMMMCMXCIX**. Si on avait à le décortiquer, cela donnerait :

MMMM	CM	XC	IX
4000 = 1000 + 1000 +1000 +1000	900 = 1000 - 100	90 = 100 - 10	9 = 10 - 1

Voici d'autres exemples :

DCCXXII = 722		
DCC	**XX**	**II**
700 = 500 + 100 + 100	20 = 10 + 10	2 = 1 + 1

MMXVIII = 2018		
MM	**X**	**VIII**
2000 = 1000 + 1000	10	8 = 5 + 1 + 1 + 1

MCMLXXII = 1972			
M	**CM**	**LXX**	**II**
1000	900 = 1000 - 100	70 = 50 + 10 + 10	2 = 1 + 1

CDXLIV = 444		
CD	**XL**	**IV**
400 = 500 - 100	40 = 50 - 10	4 = 5 - 1

Si jamais vous désirez essayez d'autres nombres, sachez que le moteur de recherche « Google » peut convertir un nombre en chiffres romains pour vous.

4 Modalités de remise

Remettez sur LÉA à l'heure et à la date indiquée par votre professeur un fichier texte comportant l'adresse vers votre dépôt Git. Cette adresse sera utilisée pour effectuer un « git clone ». Ajoutez à ce fichier tout commentaire pertinent à la correction.

Une seule journée de retard est tolérée entraînant alors une pénalité de 15 % de la note. Au-delà de ce délai, le travail est refusé et la note « 0 » est attribuée.

5 Évaluation

Le travail sera évalué selon les critères suivants. Des points seront perdus en fonction de la gravité de la faute, de la qualité globale du travail et de l'effort mis par chaque membre de l'équipe dans ce travail.

Éléments
Fonctionnalités (incluant, mais sans s'y limiter) : • Conversion de chiffres en chiffres romains. • Impossibilité de convertir un nombre négatif ou égal à 0. • Impossibilité de convertir un nombre plus grand que 4999.
Interface utilisateur (incluant, mais sans s'y limiter) : • Fichiers XML d'interface propres et bien structurés. • Visuel de l'interface utilisateur propre et stable. • Changement d'orientation et d'application supporté, sans perte de données. • Présence de messages d'erreurs clairs, avec solution, lorsque nécessaires.
Documentation du code (incluant, mais sans s'y limiter) : • Documentation convenable de la couche modèle avec de la « JavaDoc ».
Qualité générale du code (incluant, mais sans s'y limiter) : • Logique bien pensée, juste, et « non patché ». • Découpage adéquat du code en classes, méthodes et packages. • Séparation raisonnable des responsabilités entre les classes. • Respects des standards du langage de programmation. • Nommage clair et sans ambiguïté des éléments. • Utilisation de constantes, lorsque nécessaires. • Commentaires compensant uniquement le manque d'expressivité du code. • Respect des bonnes pratiques de programmation générales. • Code propre, sans résidus et facilement lisible. • Aucune erreur ni avertissement à la compilation.
Qualité générale du travail (incluant, mais sans s'y limiter) : • Configuration Gradle fonctionnelle. • Respect des consignes de remise. • Aucun fichier inutile remis avec le projet. • Travail remis via l'outil gestion de code source Git.

Notez que la qualité écrite de la langue française fait partie intégrante de l'évaluation de ce travail. Étant donné que l'accès aux ressources de la langue française est permis, chaque faute de français retirera donc 0,5 % à la note finale jusqu'à concurrence de 20 % retirés. Sont corrigées les fautes d'orthographe, les fautes de grammaire, les fautes syntaxiques et les fautes de ponctuation.