Cégep de Sainte-Foy – Automne 2024
Introduction à la programmation– 420-W10-SF

# Travail Pratique 1

Préparé par
Karine Filiatreault et Pierre Poulin
Modifié par
Raphaël Paradis

# 1 Résumé

Pour ce travail pratique, nous allons concevoir une application gérant la charge d'une batterie.

# 2 Conditions de réalisation

| Valeur de la note finale | Type | Durée | Nombre de remises |
|---|---|---|---|
| 10 % | Individuel | 6 jours | 1 |

# 3 Objectifs du travail

L'objectif de ce travail est de simuler une batterie dans une fenêtre « console ».

L'utilisateur doit d'abord entrer le numéro de série de la batterie et son numéro d'identification (ID) pour s'authentifier.

Une fois authentifié, l'utilisateur a le choix entre deux opérations : utiliser et recharger.

Un des objectifs de ce travail concerne la validation des données entrées par l'utilisateur. Par conséquent, **vous devez vous assurer que votre programme ne plante jamais** peu importe les données saisies lors de l'exécution.

# 4 Spécifications

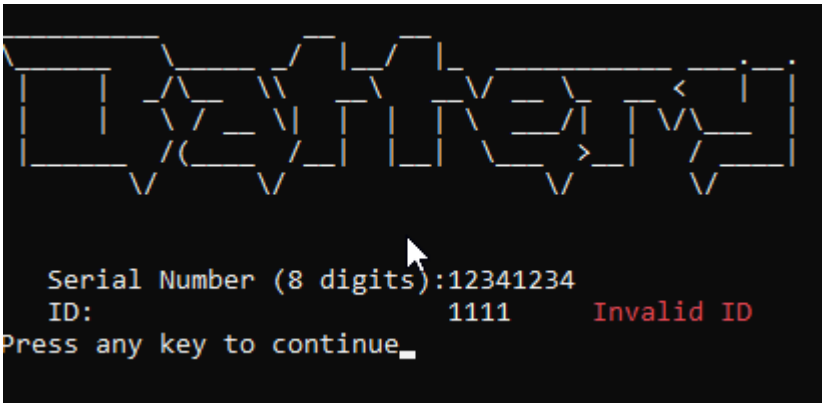
Vous devez d'abord proposer un écran d'accueil à l'usager.


L'utilisateur doit d'abord entrer le numéro de série. Ce numéro doit comporter **exactement 8 chiffres** pour être valide.

L'utilisateur doit ensuite saisir l'ID. Ce dernier doit être constitué des **4 derniers chiffres du numéro de série** pour être valide.

Si le numéro de carte est invalide, l'utilisateur peut en entrer un autre :

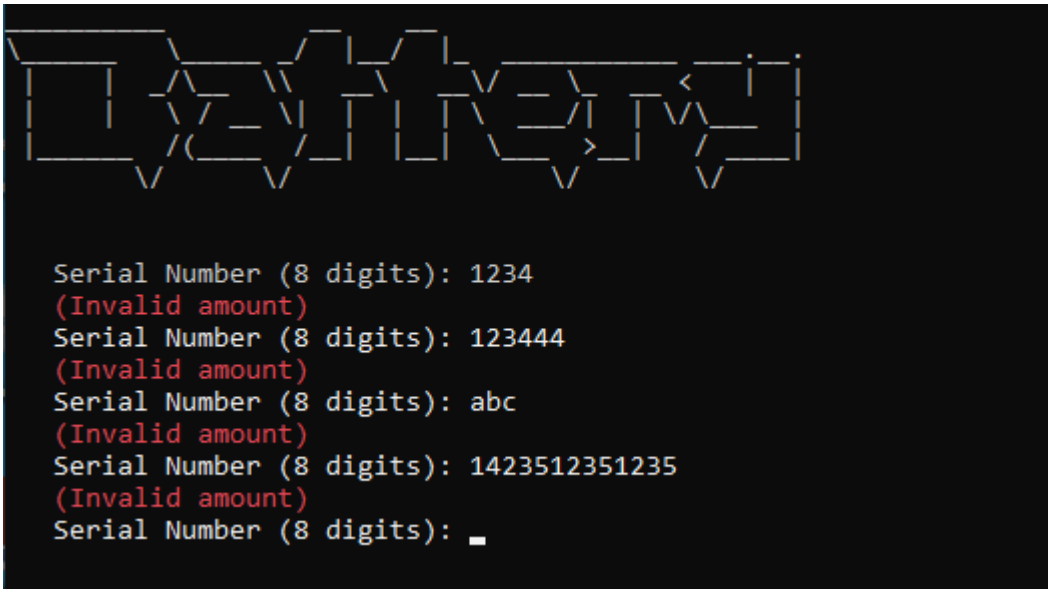

Si l'ID est invalide, l'utilisateur doit saisir à nouveau le numéro de série et l'ID.




L'important ici est qu'un message d'erreur soit affiché à l'utilisateur.

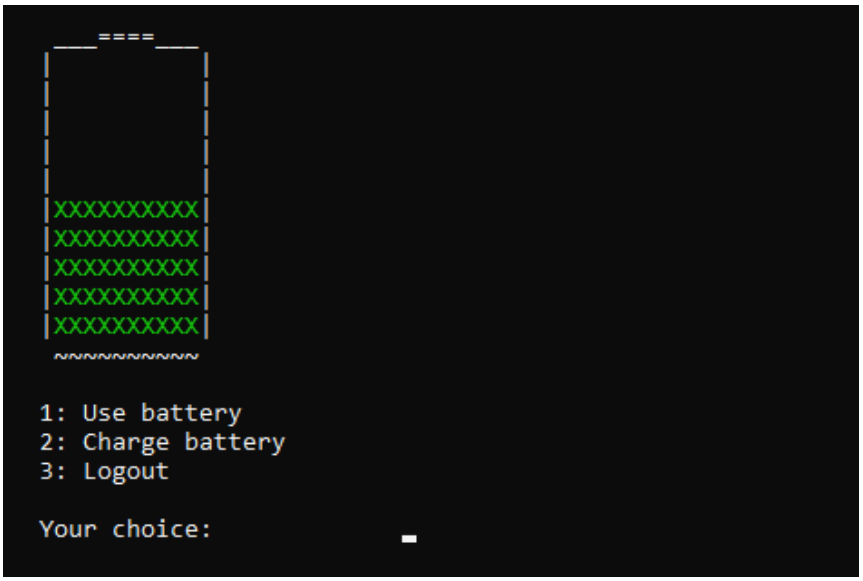
Afin d'attirer l'attention de l'utilisateur sur les messages d'erreur, ces derniers doivent être affichés en rouge à l'écran. Référez-vous au besoin à l'**Annexe 2** pour connaître la fonction permettant de modifier la couleur d'affichage.

Vous pouvez aussi choisir une approche dans laquelle les messages sont affichés à la suite :

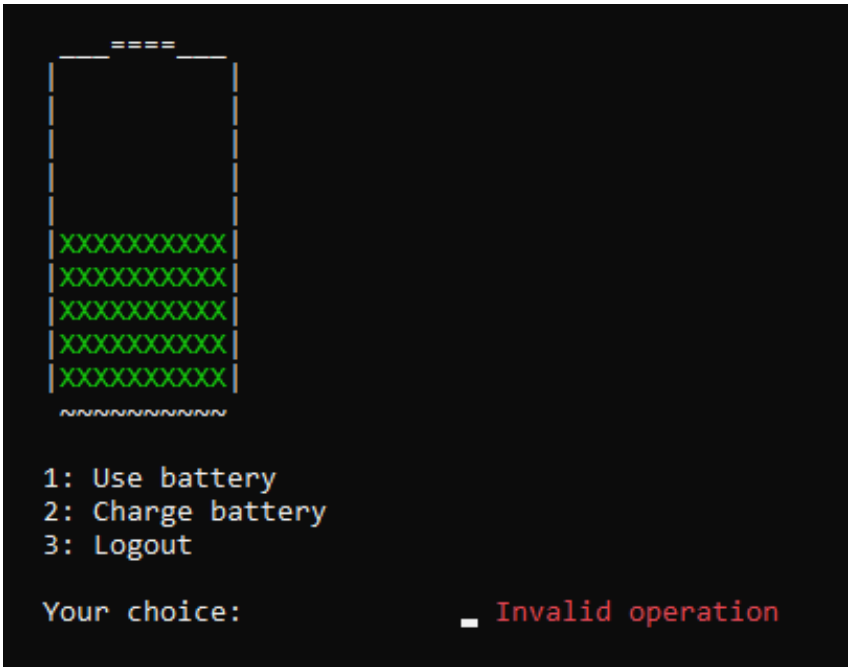


Une fois authentifié, l'utilisateur est dirigé vers la fenêtre d'opérations. A noter qu'il n'y a pas de persistance de la charge dans l'application. Dès qu'un utilisateur s'authentifie, on lui accorde 50% de charge et ce, sans égard aux opérations qu'il pourrait avoir réalisées précédemment.

La charge de la batterie doit être préservé entre les opérations jusqu'à ce que l'utilisateur quitte et retourne au menu principal.



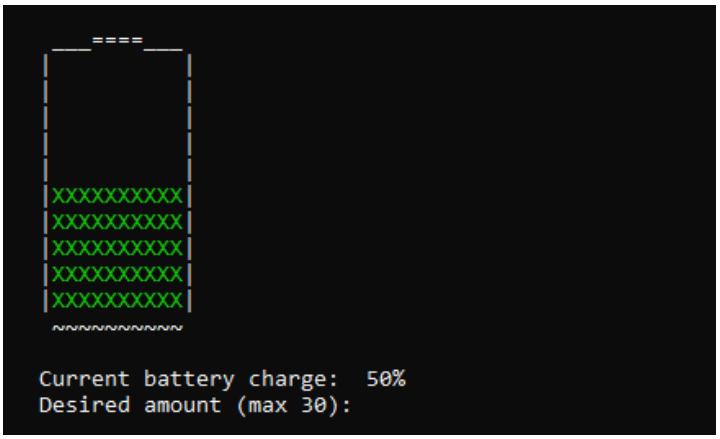

L'utilisateur peut alors saisir l'opération voulue. Seulement les chiffres 1, 2 et 3 sont acceptés. Si l'utilisateur choisit une autre option, il sera alors invité à saisir à nouveau l'opération voulue :



Encore ici, vous pouvez choisir d'afficher le message d'erreur différemment :

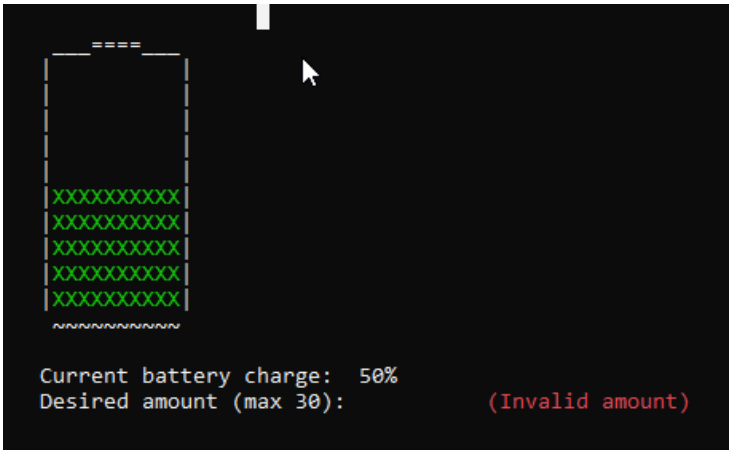
```
Your choice: a
Invalid choice
Your choice: -1
Invalid choice
Your choice: 4
Invalid choice
Your choice:
```

Si l'utilisateur entre 1 la fenêtre d'utilisation de la batterie apparaît :




L'utilisateur est alors appelé à saisir la charge qu'il souhaite utiliser. Ce montant ne peut être supérieur au plus petit des deux valeurs suivantes : la charge courante ou 30%.

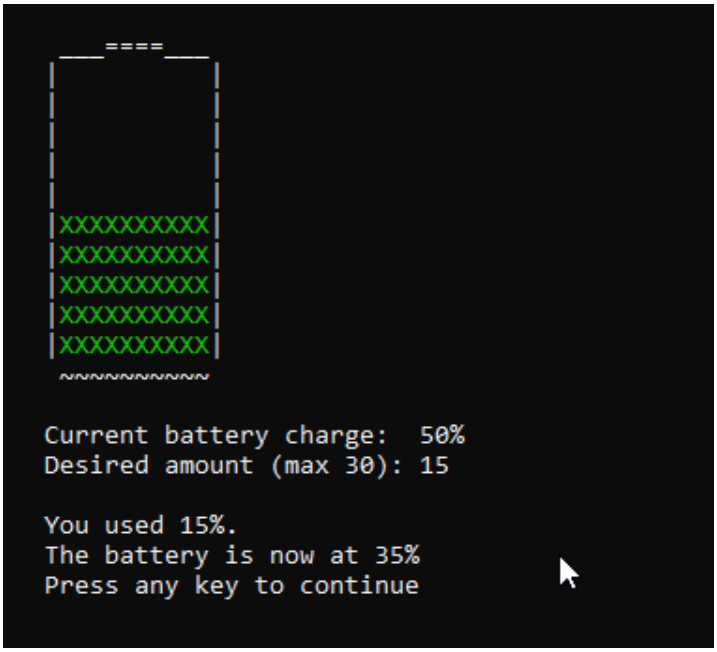
L'entrée d'un montant invalide (ne respecte pas la limite, négatif ou contenant des caractères autres que des chiffres) provoque l'affichage d'un message d'erreur et l'utilisateur doit alors entrer à nouveau le montant :




ou

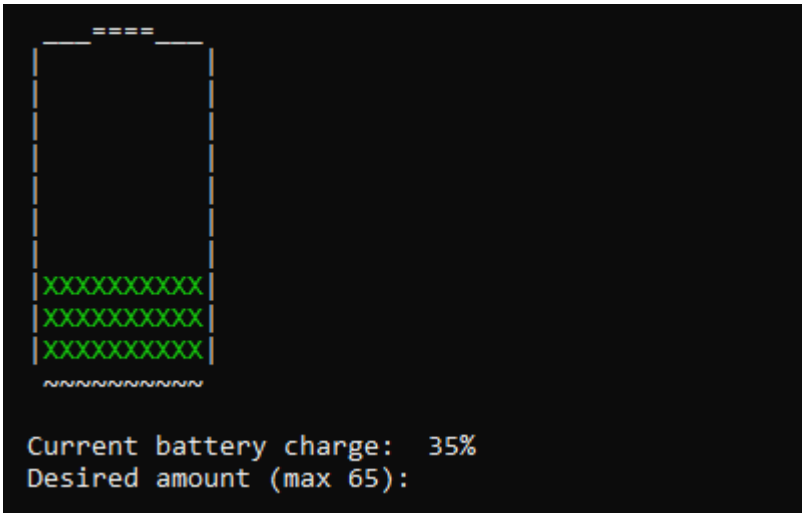
```
Current battery charge:  100%
Desired amount (max 30): 120
(Invalid amount)
Desired amount (max 30): -30
(Invalid amount)
Desired amount (max 30): 15
```

Après avoir saisi un montant valide pour l'utilisation, une confirmation est affichée à l'utilisateur :

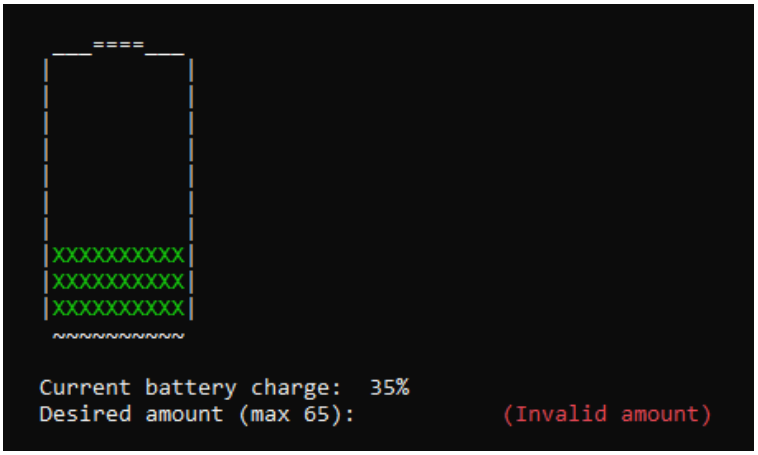



L'utilisateur appuie alors sur une touche pour retourner au menu des opérations.

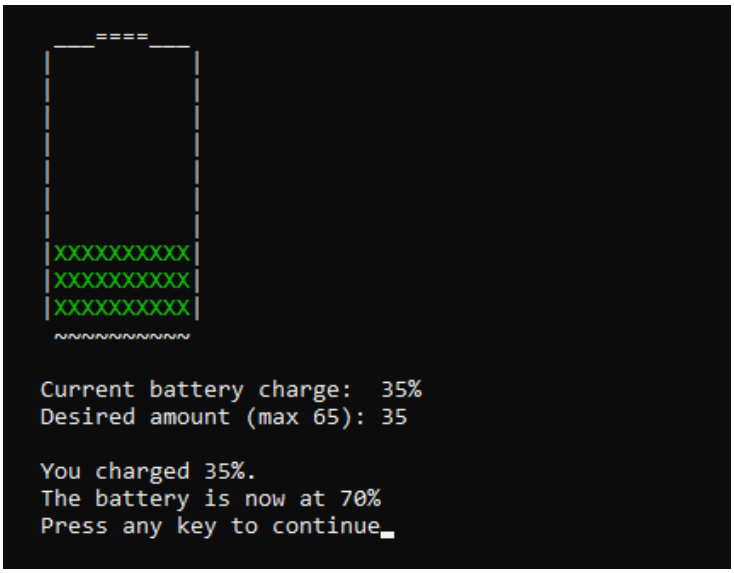
La fenêtre de recharge fonctionne de manière similaire. L'utilisateur entre la valeur souhaitée. La limite est la capacité maximale (100%) de la batterie moins la charge courante :



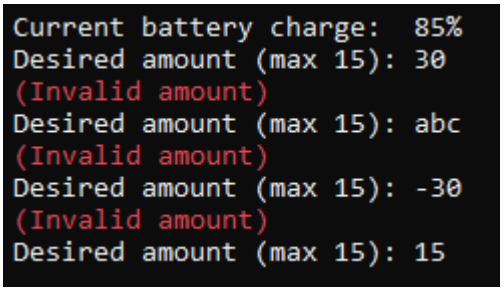
Si l'utilisateur entre une valeur invalide, il est appelé à le saisir à nouveau :




Lorsque l'utilisateur entre un montant valide, une fenêtre de confirmation est affichée :




Encore ici, il est possible de signaler les entrées erronées différemment :




### Retour à la fenêtre d'authentification

L'utilisateur qui entre 3 à la fenêtre de choix d'opération est retourné à la fenêtre d'authentification. Cette fenêtre ne peut jamais être quittée.

## 5 Contraintes de réalisation et astuces de développement

L'objectif premier de ce travail est de vous faire travailler avec les **alternatives** et les **répétitives** ainsi que de vous faire **créer et utiliser des fonctions**. Assurez-vous de découper votre code de manière à éviter de dupliquer des instructions lorsque cela n'est pas nécessaire.

Vous devez le réaliser **seul**.

N'essayez pas de tout faire en même temps.

Commencez par assumer que l'utilisateur entre toujours des données valides. Vous ajouterez les validations requises plus tard.

Vous trouverez à l'Annexe 1 un diagramme hiérarchique expliquant les appels de fonctions possibles pour le travail.

Vous trouverez à l'**Annexe 2** une liste de fonctions pouvant être utiles/pertinentes pour la réalisation de ce travail.

## 6 Évaluation

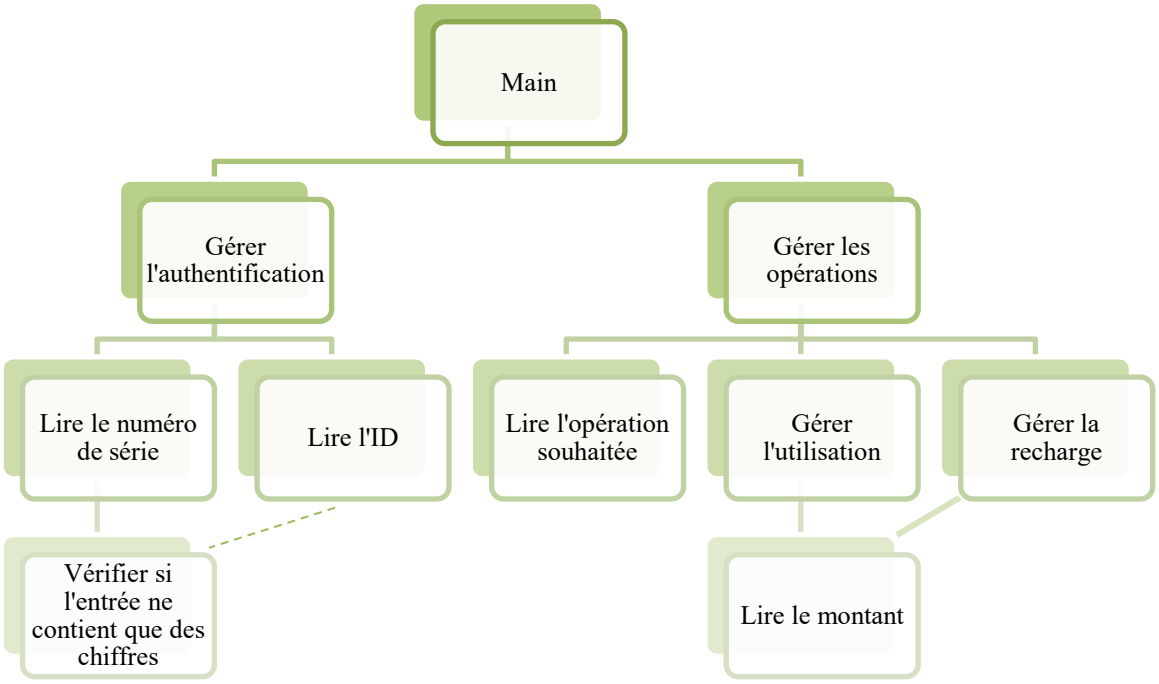
Vous trouverez dans le fichier ci-joint la grille d'évaluation utilisée pour la correction de ce travail pratique.

## 7 Modalités de remise

Remettez votre projet C# sur LÉA, dans la section travaux, à l'intérieur d'une archive *Zip*. Supprimez tous les fichiers et dossiers temporaires, à savoir les dossiers `.vs`, `bin` et `obj`.

# Annexe 1

## Diagramme hiérarchique du projet



# Annexe 2

Fonctions qui peuvent être utiles pour la réalisation du travail

| Fonctions | Descriptions |
|---|---|
| `Console.Clear();` | Permet d'effacer le contenu de la console |
| `Console.SetCursorPosition(int colonne, int ligne);` | Permet de positionner le curseur à une position spécifique de la console.<br><br>Note : la ligne 0 est la première de la console. Plus le numéro de ligne est élevé, plus la ligne est basse dans la console. |
| `Console.ForegroundColor = ConsoleColor.XYZ;` | Permet de changer la couleur du texte écrit dans la console. La couleur reste la même jusqu'à ce qu'elle soit changée à nouveau. Les valeurs possibles sont de types `ConsoleColor`. Par exemple,<br>`Console.ForegroundColor = ConsoleColor.Red;`<br>permet d'écrire en rouge. |
| `Console.ReadKey();` | Permet de lire le prochain caractère sans attendre que l'utilisateur appuie sur **z**. Cette fonction peut être utile dans le cas où l'on attend que l'utilisateur appuie sur une touche pour continuer. |
| `char.IsNumber(char caractere);`<br><br>ou<br><br>`char.IsDigit(char charactere);` | Permet de tester si **caractere** est un chiffre. Par exemple, pour le code suivant :<br><br>`char caract1 = '2';`<br>`char caract2 = 'a';`<br>`bool isCaract1ADigit = Char.IsNumber(caract1);`<br>`bool isCaract2ADigit = Char.IsNumber(caract2);`<br>**isCaract1ADigit** et **isCaract2ADigit** vaudraient respectivement **true** et **false**. |

Le site suivant peut aussi vous aider pour générer du texte pour les titres. C'est celui utilisé pour générer les maquettes présentées dans cet énoncé.