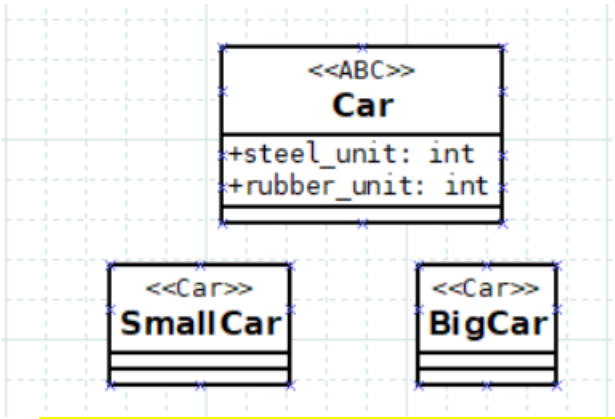
Ce diagramme décrit un modèle permettant de représenter 2 types d'automobile petite ou grande ainsi que le nombre d'unités d'acier ou de caoutchouc nécessaire à leur production; une petite auto requiert 1 unité d'acier et 1 unité de caoutchouc alors qu'une grosse auto requiert 1 unité de caoutchouc et 2 unités d'acier.



1. Dans un fichier car.py, définissez la classe parent abstraite « Car » avec un constructeur paramétré qui initialise les 2 attributs.
2. Dans un fichier small_car.py, définissez la classe « SmallCar » (enfant de Car) avec un constructeur permettant d'initialiser cette classe sans paramètre; en utilisant le constructeur du parent vous devez initialiser les bonnes quantités d'acier et de caoutchouc pour une petite auto (1 unité d'acier et 1 unité de caoutchouc).
3. Dans un fichier main.py, instancier la classe « SmallCar » puis, avec un print(…), affichez ses attributs.

```
Q3) 1 1
```

4. **Sur le même principe, définissez la classe « BigCar » avec un constructeur sans paramètre initialisant les bonnes quantités d'acier et de caoutchouc.**
   Dans le fichier main.py, instancier la classe « BigCar » puis affichez ses attributs.

```
Q4) 2 1
```

5. Ajoutez une méthode fournissant une représentation textuelle d'une instance de petite auto (__str__). Voici le texte à reproduire;
   Testez cette méthode dans votre fichier main.py

```
Q5) Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
```

6. **Sur le même principe, ajoutez une méthode fournissant une représentation textuelle d'une instance de grande auto.**
   Testez cette méthode dans votre fichier main.py

```
Q6) Je suis une auto de grande taille. Je consomme 2 unité(s) d'acier et 1 unité(s) de caoutchouc
```

7. Appliquez le pattern constant pour représenter les quantités d'acier et de caoutchouc nécessaire à la production des 2 types d'autos.
8. Créez un dictionnaire pour conserver les 2 instances d'auto créées à la question 3 et 4 avec 2 clefs "c1" pour la petite auto et "c2" pour la grande.
9. Parcourez le dictionnaire pour afficher chacune de ses clefs avec sa valeur;

```
Q9) c1:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
Q9) c2:Je suis une auto de grande taille. Je consomme 2 unité(s) d'acier et 1 unité(s) de caoutchouc
```

10. Ajoutez au dictionnaire une clef "c3" contenant une nouvelle instance de petite auto.
11. Créez une **variable** c3 pour obtenir la valeur du dictionnaire associée à la clef "c3".

12. Dans le dictionnaire, remplacez la valeur associée à la clef "c2" par le contenu de la **variable** c3
13. Supprimez la clef "c3" du dictionnaire.
    Parcourez le dictionnaire pour afficher chacune de ses clefs avec leur valeur;

```
Q13) c1:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
Q13) c2:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
```

14. Typez le code du fichier car.py
15. Typez le code du fichier main.py
16. **À l'aide d'un décorateur, loguez (en utilisant print(…)) chaque création d'instance d'auto en indiquant le type de l'instance (petite ou grande) et les quantités d'acier et de caoutchouc comme ci-après.**

```
16) Une nouvelle auto a été créée de type <class 'small_car.SmallCar'> acier = 1 caoutchouc = 1
```

17. Déposez sur Léa un seul fichier archive contenant votre code Python avec un fichier par classe et aucun code généré par Pycharm.

Voici un exemple de résultat final du fichier main.py; le numéro au début de la ligne fait référence à la question correspondante

```
16) Une nouvelle auto a été créée de type <class 'small_car.SmallCar'> acier = 1 caoutchouc = 1
Q3) 1 1
16) Une nouvelle auto a été créée de type <class 'big_car.BigCar'> acier = 2 caoutchouc = 1
Q4) 2 1
Q5) Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
Q6) Je suis une auto de grande taille. Je consomme 2 unité(s) d'acier et 1 unité(s) de caoutchouc
Q9) c1:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
Q9) c2:Je suis une auto de grande taille. Je consomme 2 unité(s) d'acier et 1 unité(s) de caoutchouc
16) Une nouvelle auto a été créée de type <class 'small_car.SmallCar'> acier = 1 caoutchouc = 1
Q13) c1:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
Q13) c2:Je suis une auto de petite taille. Je consomme 1 unité(s) d'acier et 1 unité(s) de caoutchouc
```

Barème; toutes les questions valent 5 points, exceptées les questions 4 et 6 et 16 qui valent 10 points. Vous perdez de 10 à 30 % des points selon que vous respectez ou non les contraintes de code de la question. Par exemple la question 1 stipule 2 contraintes de code; la classe doit être abstraite et le constructeur paramétré.

Vous devez déposer 2 versions de votre code (chacun dans un fichier zip différent);

1. Une première version avant la fin de la durée de l'examen en classe. Elle compte pour 30 % de la note finale. Aucune documentation autorisée exceptée les notes et le code vu en classe.
    - Vous obtenez 0 points si vous utilisez du code pas vu en classe.

2. Une seconde version avant la date limite indiquée sur Léa qui compte pour 70% de la note finale. Toute documentation autorisée (y compris chatGPT). Pour cette seconde version uniquement;
    - Si vous utilisez internet (Google ou ChatGPT), fournissez votre référence et votre auto-évaluation en déposant ce document;
        - Surlignez en jaune les questions auxquelles vous avez répondu uniquement à l'aide des notes de cours, sans l'aide d'internet,
        - En vert celles pour lesquelles vous avez eu de l'aide mais que vous avez comprises (vous pourriez les expliquer et/ou les refaire sans aide d'internet à partir d'un exemple)
        - Et en rouge les questions non maîtrisées (vous ne pourriez pas les expliquer et/ou les refaire sans aide d'internet). Vous obtiendrez que 10% des points pour les questions surlignées en rouge.
    - Vous obtenez 0 points si vous utilisez du code pas vu en classe.
    - Vous obtenez 0 points si votre code ne compile pas (erreur avec le code couleur rouge dans Pycharm, commentez-le pour obtenir des points).
    - Vous perdez de 1 à 2 points par fichier selon le nombre d'avertissements concernant les conventions PEP8.
    - Question 18) Bonus de 10 points - exception personnalisée;
        - Modifier le constructeur de SmallCar et de BigCar pour qu'il accepte des paramètres afin d'initialiser les quantités d'acier et de caoutchouc avec des valeurs par défaut au cas où les paramètres ne sont pas utilisées. Voici un exemple de constructeur paramétré avec valeur par défaut;

```python
class Personne():
    #Constructeur paramétré avec valeur par défaut
    def __init__(self, nom:str = "nom par défaut"):
        self.nom = nom
p1: Personne = Personne()
p2: Personne = Personne("Martin")
print(f"Nom de p1 = {p1.nom}, Nom de p2 = {p2.nom}")
# Nom de p1 = nom par défaut, Nom de p2 = Martin

```

        - Ajouter une exception personnalisée CarUnitException qui doit être lancée si une auto (petite ou grande) est créée avec une quantité qui n'est pas comprise entre 1 et 3 (bornes incluses)
        - Tester plusieurs cas d'exception; voici un exemple de message à afficher si on instancie une auto avec de mauvaise valeur.

```
Cette quantité d'acier 0 n'est pas valide et doit être comprise entre 1 et 3
Cette quantité d'acier 4 n'est pas valide et doit être comprise entre 1 et 3
```

        - Combien de tests devriez vous faire pour couvrir tous les cas possibles d'instanciation d'auto incorrect?
        - Pour obtenir tous les points votre code doit éviter que du code soit répété inutilement.