La classe **Robot** modélise l'état et le comportement de robots virtuels. Chaque robot correspond à un objet qui est une instance de cette classe.

Dans les développements demandés, testez les au fur et à mesure que vous les réalisez dans une classe **Main**.

Chaque robot :
• a un nom (attribut **nom** : chaîne de caractères)
• a une position : donnée par les attributs entiers **x** et **y**, sachant que **x** augmente en allant vers l'Est et **y** augmente en allant vers le Nord
• a une direction : donnée par l'attribut **direction** qui prend une des valeurs **"Nord"**, **"Est"**, **"Sud"** ou **"Ouest"**
• peut avancer d'un pas en avant : avec la méthode sans paramètre **avance()**
• peut tourner à droite de 90° pour changer de direction (si sa direction était "Nord" elle devient "Est", si c'était "Est" elle devient "Sud", etc.) : avec la méthode sans paramètre **droite()**. Les robots ne peuvent pas tourner à gauche.
• peut **afficher** son état en détail (avec de simples **System.out.println()**)

Le nom, la position et la direction d'un robot lui sont donnés au moment de sa création. Le nom est obligatoire mais on peut ne pas spécifier la position et la direction, qui sont définis par défaut à **(0,0)** et **"Est"**.

1) Écrire les instructions Java qui permettent de définir la classe **Robot**, **en respectant le principe de l'encapsulation des données**.

2) On veut améliorer ces robots en en créant une Nouvelle Génération, les **RobotNG** qui ne remplacent pas les anciens robots mais peuvent cohabiter avec eux.

Les **RobotNG** savent faire la même chose mais aussi :
• avancer de plusieurs pas en une seule fois grâce à une méthode **avance()** qui prend en paramètre le nombre de pas
• tourner à gauche de 90° grâce à la méthode **gauche()**
• faire demi-tour grâce à la méthode **demiTour()**

Écrire cette nouvelle classe en **spécialisant** celle de la première question, **sans modifier celle-ci**:

a) dans un 1er temps, les nouvelles méthodes appellent les anciennes méthodes pour implémenter le nouveau comportement : avancer de *n* pas se fait en avançant de 1 pas *n* fois, « tourner à gauche » se fait en tournant 3 fois à droite, faire demi-tour se fait en tournant 2 fois

b) donner une 2nde solution plus efficace qui change directement l'état de l'objet sans faire appel aux anciennes méthodes (...mais attention aux droits d'accès !); cette seconde approche devra « cohabiter » avec la première …

3) On veut mettre ensemble dans une classe **ListeDeRobots** des objets de type **Robot** et de type **RobotNG**.

Définir cette classe **ListeDeRobots**, en fournissant en particulier les méthodes suivantes :

- ajouter(Robot r) qui ajoute un **Robot** ou un **RobotNG** à une instance de cette classe ;
- afficher l'état de tous les robots contenus dans une instance de cette classe ;
- afficher l'état des seuls robots de type **RobotNG** contenus dans une instance de cette classe.

4) Une évolution de la classe **RobotNG** introduit un mode « Turbo » qui sera par défaut inactif, mais qui pourra être activé à la demande (et donc également désactivé). Dans le mode actif, chaque pas du robot est multiplié par 3. L'appel à la méthode **afficher()** devra pour sa part indiquer à la fin si le robot est en mode Turbo ou pas.

Proposer une implémentation de cette évolution …