Generics, hiérarchie de types

Exercice 1 - Pile suite

1 Écrire une autre classe de pile, `fr.umlv.util.stack.LinkedStack`, possèdant les méthodes `isEmpty`, `push` et `pop` et utilisant une `java.util.LinkedList` comme structure de stockage sous-jacente.

2 Indiquer dans quels cas, il est mieux d'utiliser une `FixedStack` ou une `LinkedStack`.
Comment doit-on faire s'il l'on veut pouvoir dynamiquement choisir entre les deux implantations pour exécuter un même code ?
Modifier votre code et écrire un `main` de test.

3 Générifier l'ensemble des classes crées.

Exercice 2 - Minimum (plus)

On souhaiterait écrire une nouvelle méthode `min` qui puisse être aussi utilisée avec des chaînes de caractères ou tout autre objet comparable (regarder `java.util.Comparable`).

```
System.out.println(min(3,12));             // affiche 3
System.out.println(min("trois","douze")); // affiche douze
```

1 Qu'elle est la signature de la méthode `min` ?

2 Ecrire le code correspondant

Exercice 3 - Pile suite suite

1 Expliquer la différence entre `java.util.ArrayList` et `java.util.LinkedList`.

2 Ajouter une nouvelle implantation de `Stack` utilisant une `java.util.ArrayList` nommé `fr.umlv.util.stack.ArrayStack`.

3 Rappeler pourquoi il est essentiel de ne pas avoir le même code écrit plusieurs fois. (Plusieurs réponses sont possibles !)

4 On rappelle que `java.util.ArrayList` et `java.util.LinkedList` implante la même interface `List`, il est donc possible de partager du code entre deux des implantations de `Stack`, `fr.umlv.util.stack.LinkedStack` et `fr.umlv.util.stack.ArrayStack`.
Où doit-on mettre le code commun ?

5 Changer le code pour qu'il soit partagé au maximum.
Attention on veux garder une complexité optimale des opérations et éviter les casts inutiles.