Héritage

Objectif

L'objectif de ce TP est de développer un système de classes permettant de générer des nombres aléatoires et des processus stochastiques. Une étude détaillée des différents mécanismes mis en jeu devra permettre de mettre en évidence une hiérarchie de classe : une classe mère `GenerateurNombreAleatoire` de laquelle les différentes classes de générateurs dépendront.

Le second système de classe utilisé sera consacré à la définition des distributions statistiques par héritage de la classe `Distribution`.

Les notions abordées dans ce TP sont : l'héritage , l'encapsulation, la composition et l'agrégation.

Vous devrez apporter un soin tout particulier à la gestion de la mémoire et à la documentation de votre code. La gestion des erreurs doit se faire via des exceptions (utilisez `throw`).

Implémentation

1. Construire une classe `GenerateurNombreAleatoire` abstraite qui servira de base aux autres générateurs. Cette classe ne contiendra qu'un seul attribut, qui est la dimensionnalité du générateur.
   La classe `GenerateurNombreAleatoire` devra proposer les fonctionnalités suivantes :
   - Les 4 méthodes de base d'une classe en identifiant celles qui doivent être désactivées, c'est-à-dire en exposant uniquement celles utilisables par un autre programme.
   - Une méthode de clonage :`clone`.
   - Des accesseurs pour la dimensionnalité.
   - Des méthodes de manipulation de la graine du générateur : `set_seed`, `get_seed` et `reset_seed`.
   - Une méthode de génération de nombre générés uniformément dans un tableau.

   Chaque générateur ayant des structures et paramètres différents, la classe mère ne contient pas de méthode génératrice directement. Les classes filles posséderont donc un membre agrégé qui s'occupera de la génération.
2. Le premier générateur que nous allons implémenter est celui de Park Miller. C'est un générateur par congruence linéaire. La classe `ParkMiller` devra contenir les éléments suivants
   - Les 4 méthodes de base d'une classe en identifiant celles qui doivent être désactivées, c'est-à-dire en exposant uniquement celles utilisables par un autre programme.
   - Un attribut seed uniquement accessible par les accesseurs.
   - Une méthode de génération d'un entier aléatoire.

   La génération de nombre aléatoire par la méthode de Park Miller utilise 4 nombres entiers définit dans le document descriptif général.

Remarque 1

On devra prêter particulièrement attention à la déclaration de ces entiers.

3. Afin d'utiliser ce générateur depuis la classe `GenerateurNombreAleatoire`, il est nécessaire de dériver cette classe. Faire hériter une classe `GenerateurParkMiller` depuis cette classe mère.
4. Renouveler les étapes précédentes avec le générateur XorShift.

La seconde classe générique que nus allons définir concerne les distributions statistiques. Pour cela, nous allons définir une classe mère `Distribution`.

5. La classe devra contenir les 4 méthodes de bases.
6. Une méthode de transformation d'une distribution uniforme en une autre distribution `random_draws`.
7. Les méthodes `cdf`, `pdf`, `mean`, `var`, `stdev` et `inv_cdf`.
8. Définir une classe `DistributionNormale` qui construit une distribution normale à partir d'une distribution uniforme.

Analyse

Question 1

Pour chacune des classes implémentées, définir des tests pertinents.

Question 2

Évaluer les performances des générateurs aléatoires implémentés.

Question 3

Comment mettre en évidence les forces et faiblesses des générateurs proposés.