SEQUENCE 4 – STRUCTURES REPETITIVES
SEANCE 4 - STRUCTURES CONDTIONNELLES, REPETITIVES ET METHODES GESTION DE ERREURS

OBJECTIFS

- Revoir les tests et les boucles au sein des méthodes
- Apprendre à faire une double vérification :
  - Gérer les cas d'erreurs dans les méthodes avec de if (paramètres non valides dans les méthodes) : Exceptions pour le programmeur
  - Gérer les cas d'erreurs dans le main avec des boucles : Messages pour l'utilisateur

Remarque : si les vérifications sont bien faites dans le main, cela n'engendra pas d'exception. Pour autant, il faut tout de même prévoir les cas d'erreurs.

EXERCICE 1 : BANQUE

1. Créez un projet **« Exo1_CreationCompteBancaire »** dans une solution nommée « **Seance4_GestionsErreursMainEtMethodes»** dans votre répertoire «**Sequence_4_StructuresRepetitives**».

2. Copiez/collez ces lignes dans le main. **Remarque : ici, on utilise la méthode avec des données fixes pour la tester, pas besoin de saisies utilisateur dans un 1er temps.**

```
class Program
   {
    public readonly double TARIF_IMMEDIAT = 31.5;
    public readonly double TARIF_DIFFERE = 42.5;
    public readonly double TAUX_PREFERENTIEL_JEUNE = 0.5;
    public readonly int  AGE_PREFERENTIEL = 25;
    public readonly int  AGE_MINIMAL = 16;


    static void Main(string[] args)
     {
       Console.WriteLine("Ouverture compte 11 ans :"
                         + Program.OuvertureCompteCheque(11));
       Console.WriteLine("Ouverture compte 18 ans :"
                         + Program.OuvertureCompteCheque(18));
    //   Console.WriteLine("Ouverture compte -8 ans :"
    //                      + Program.OuvertureCompteCheque(-8));
```

3. Puis définissez la méthode : `OuvertureCompteCheque`
   public static bool OuvertureCompteCheque (int age) - retourne true si l'ouverture du compte est possible, false sinon. Seule condition pour ouvrir un compte: avoir 16 ans ou plus. **Attention lancez une exception ArgumentOutOfRangeException en cas d'âge <0.**

4. **Remplissez le tableau et testez (décommentez puis re commentez le test d'erreur)**

Donnée en entrée (age)	Résultat attendu
-1	Erreur
11	False
18	True

5. Complétez le main avec les lignes suivantes :

```
Console.WriteLine("Prix différé 35 ans :" + Program. PrixCarteBleue ("D", 35));
Console.WriteLine("Prix immédiat 35 ans :" + Program. PrixCarteBleue ("I", 35));
Console.WriteLine("Prix différé 25 ans :"+Program. PrixCarteBleue ("D", 25));
Console.WriteLine("Prix immédiat 25 ans :" + Program. PrixCarteBleue ("I", 25));
// Console.WriteLine("Prix immédiat 25 ans :" + Program. PrixCarteBleue ("IM", 25));
// Console.WriteLine("Prix différé 5 ans :"+Program. PrixCarteBleue ("D", 5));
```

6. Puis définissez la méthode **:**
   public static double PrixCarteBleue (String debit , int age) - retourne le prix de la carte bleue en fonction des paramètres spécifiés et des règles énoncées ci-dessous.
   Attention, lancez des exceptions si le debit et l'age sont invalides (age >=16 pour être valide). Remplissez le tableau de jeux de tests.

   Tarif carte bleue:

   - débit immédiat : 31.50 € par an
   - débit différé : 42.50 € par an
   - tarifs préférentiels pour les jeunes :
     - 16 -25 ans : 1/2 tarif

7. **Remplissez le tableau et testez (décommentez puis re commentez les tests d'erreurs)**

Données en entrée		Résultat attendu
debit	age	prix
D	35	42.5
I	35	31.5
D	25	21.25
I	25	15.75
IM	25	Erreur débit incorrrect
D	5	Erreur trop jeune

8. Maintenant mettez le code du main qui a servi à tester en commentaire puis faites un vrai programme : il doit demander à l'utilisateur son âge. Puis s'il est possible d'ouvrir un compte, il doit demander le type débit désiré puis afficher le montant de la cotisation. Pour éviter tout lancement d'exception (et plantage), vous devez vérifier chaque saisie réutilisateur avant de déclencher la méthode : vous pouvez r**éutilisez votre dll.**

EXERCICE 2 : FACTORIELLE

Créez un projet **« Exo2_Factorielle »**
Définissez la méthode factorielle : 5 ! => 5*4*3*2*1
public static int factorielle (int nb) ;
Remarque : **le paramètre ne peut pas être négatif, déclenchez une exception si c'est le cas.**

EXERCICE 3 : POUR LES + RAPIDES : JEU DE DES – IMBRIQUER DES FONCTIONS

Vous devez : (voir impression écran page suivante)

1. **Définir :**
   - **int lanceUnDe ( )** – elle simule un lancer de dé et renvoie un chiffre compris entre 1 et 6.
   - **int nbDeUn (int nbLances)** – elle lance un dé autant de fois que l'indique le paramètre nbLances puis renvoie le nombre de un obtenu(s)

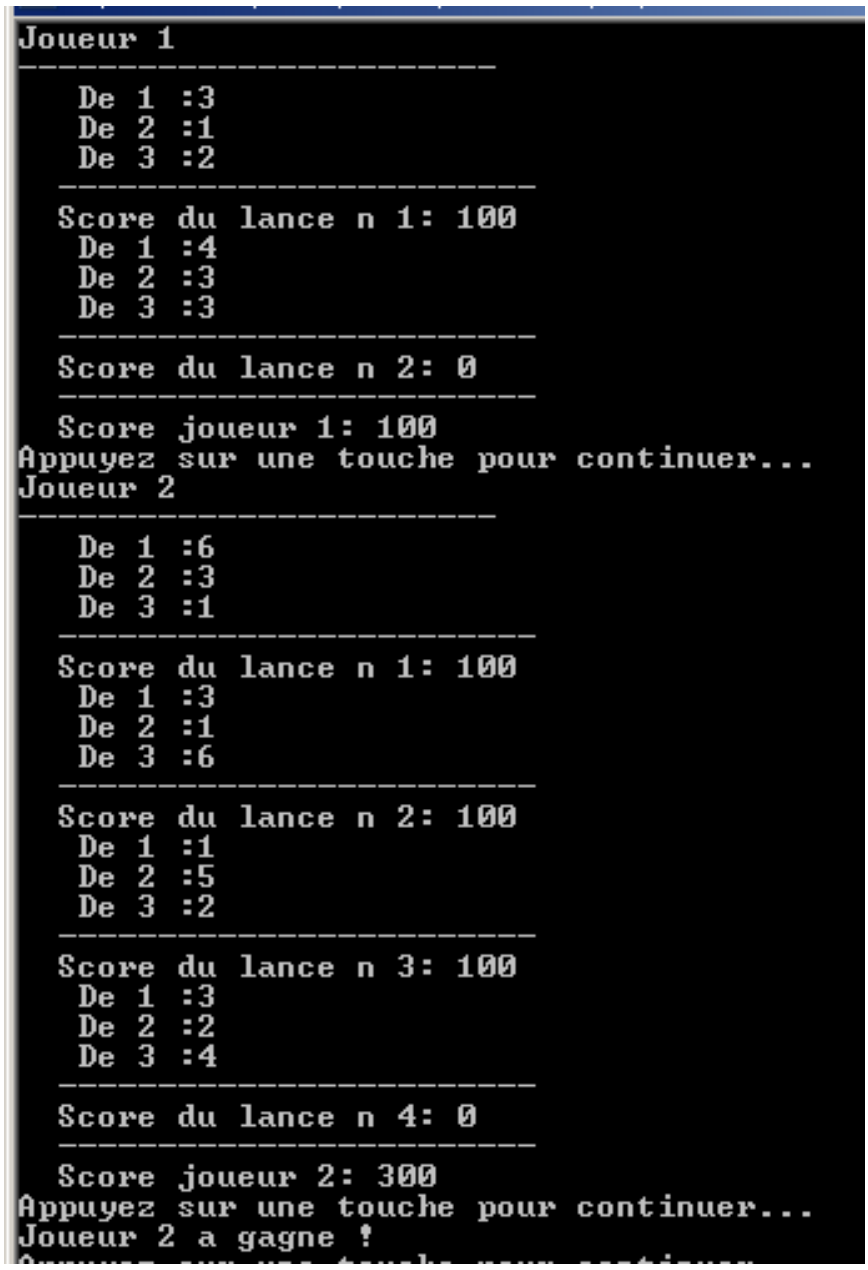
2. **Ecrire les instructions pour lancer 5 dès et afficher le nombre de 1.**

3. **Redéfinir la méthode « nbDeUn » en « nbDe » afin de la généraliser :** elle doit fonctionner pour tout dé recherché.

4. **Ecrire les instructions pour lancer 6 dès et afficher le nombre de 3.**

5. **Ecrire les instructions pour le jeu de dés suivant : Rappel des règles :**
   - 2 joueurs s'affrontent
   - le but est de dépasser les 1000 points
   - à chaque tour, un joueur lance 3 dés et voit ses dés et son score, il comptabilise 100 points par dé de 1.
   - tant qu'il réussit à avoir au moins un dé à 1, il peut continuer et relancer les dés

6. **Améliorer le programme :**
   - en ajoutant la règle suivante : lors d'un tour, le joueur doit préciser s'il désire continuer à lancer les dés, si c'est le cas et qu'il ne fait pas de un, son score est remis à 0, il perd le score des lancés précédents.
   - en rendant paramétrable le nombre de joueurs (utilisez des tableaux)



Et pour les supers rapides : essayez de faire un mode graphique !