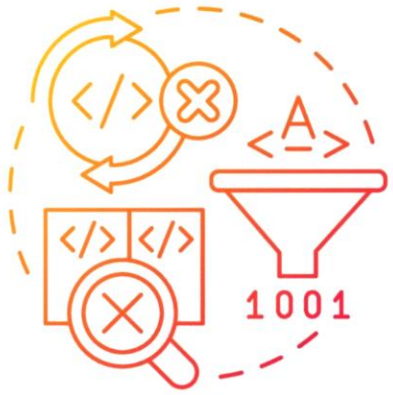
informatique



SINGLE-USE CODE

ICT320

POO

PROGRAMMATION ORIENTEE OBJET

ANASS BENFARES &
XAVIER CARREL

ANASS BENFARES

Déroulement

- Accès à la Roadmap du cours : https://roadmap.sh/r/embed?id=66714f98c0f2325c34220bba

- 5 périodes de cours avec moi et des périodes de projet avec M.Melly.
- Théorie puis exercices pratiques.
- Informations concernant les évaluations à ce lien
- Ne pas oublier, les conventions de codage ETML !

Sujets

- Exceptions
  - Try/Catch/Finally
  - Les types d’exceptions
  - Créer sa propre exception
- Tests unitaires
  - Arrange, Act, Assert
- Interface

Exceptions - Concept

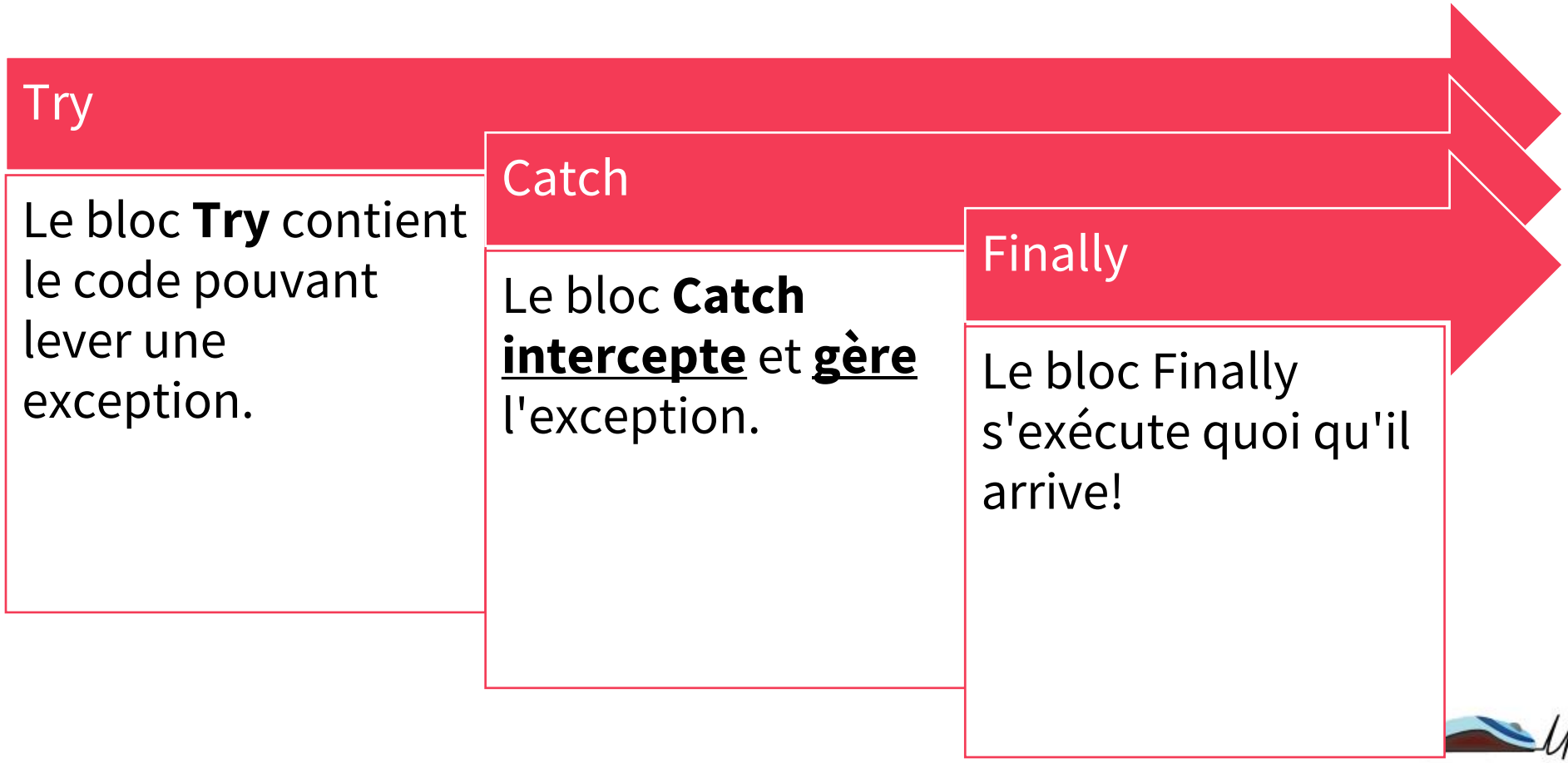
Qu'est-ce qu'une exception ?

- Une exception est une erreur qui survient lors de l'exécution du programme.
- Les exceptions interrompent le flux normal du programme.

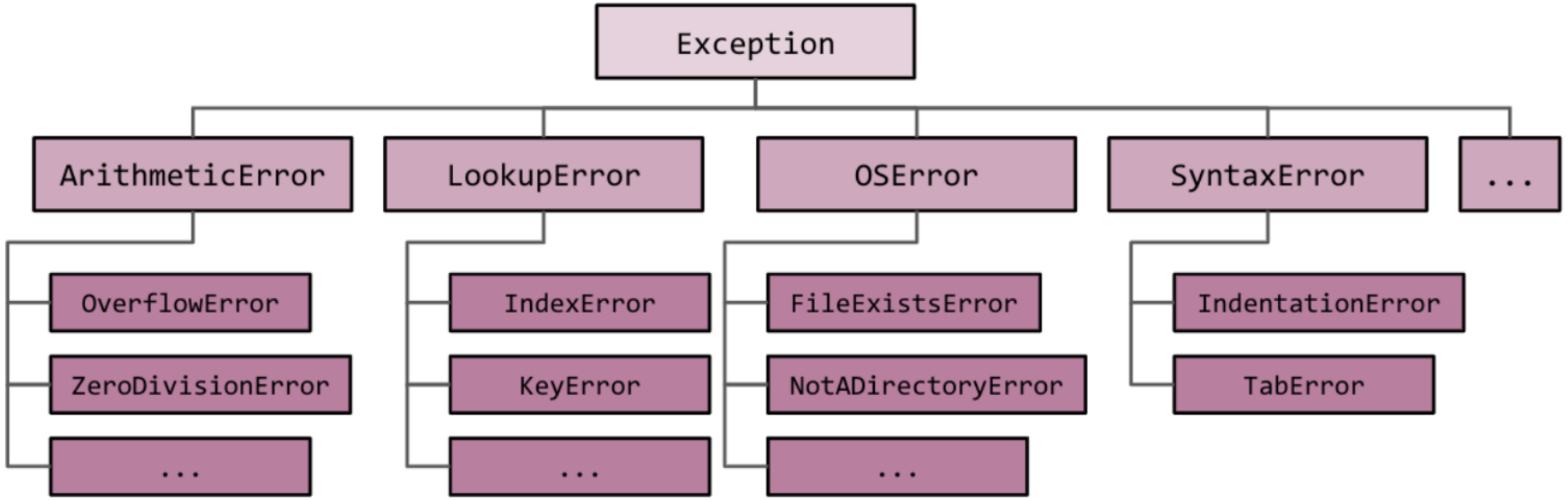
Pourquoi gérer les exceptions ?

- Pour éviter que le programme ne plante et pour gérer les erreurs de manière contrôlée.

Exceptions – Try, Catch, Finally



Exceptions – Types d'exceptions



Source : UNIL – Eléments de programmation // Cours du Mauro Cherubini

Lever une Exception / Exemple

```
Console.WriteLine("Entrez un nombre :");
string input = Console.ReadLine();

try
{
    // Essayer de convertir l'entrée en entier
    int nombre = int.Parse(input);
    Console.WriteLine($"Le nombre entré est : {nombre}");
}
catch (FormatException)
{
    // Gestion de l'exception si l'entrée n'est pas un nombre valide
    Console.WriteLine("Erreur : Vous devez entrer un nombre valide.");
}
finally
{
    // Bloc toujours exécuté
    Console.WriteLine("Opération terminée.");
```

Lever une Exception / Exemple

```
FileStream fichier = null;

try
{
    // Essayer d'ouvrir un fichier
    fichier = File.Open("chemin_vers_le_fichier.txt", FileMode.Open);
    Console.WriteLine("Fichier ouvert avec succès.");
}
catch (FileNotFoundException ex)
{
    // Gestion de l'exception si le fichier est introuvable
    Console.WriteLine("Erreur : Le fichier n'a pas été trouvé.");
}
catch (UnauthorizedAccessException ex)
{
    // Gestion de l'exception si l'accès est refusé
    Console.WriteLine("Erreur : Accès refusé au fichier.");
}
catch (Exception ex)
{
    // Gestion générale pour toute autre exception
    Console.WriteLine($"Erreur : {ex.Message}");
}
finally
{
    // Ce bloc est toujours exécuté
    if (fichier != null)
    {
        fichier.Close();
        Console.WriteLine("Fichier fermé.");
    }
}
```

Gestion d’ouverture de fichier

Créer sa propre Exception

Vous pouvez définir des exceptions spécifiques à vos besoins.

Héritage de Exception:

```
2 références
public class MonException : Exception
{
    1 référence
    public MonException(string message) : base(message) { }
}

// Utilisation
throw new MonException("Une erreur personnalisée.");
```

Visibilité

Visibilité	Classe	Classes dérivées	Extérieur
`public`	X	X	X
`protected`	X	X	
`private`	X		

Tests unitaires

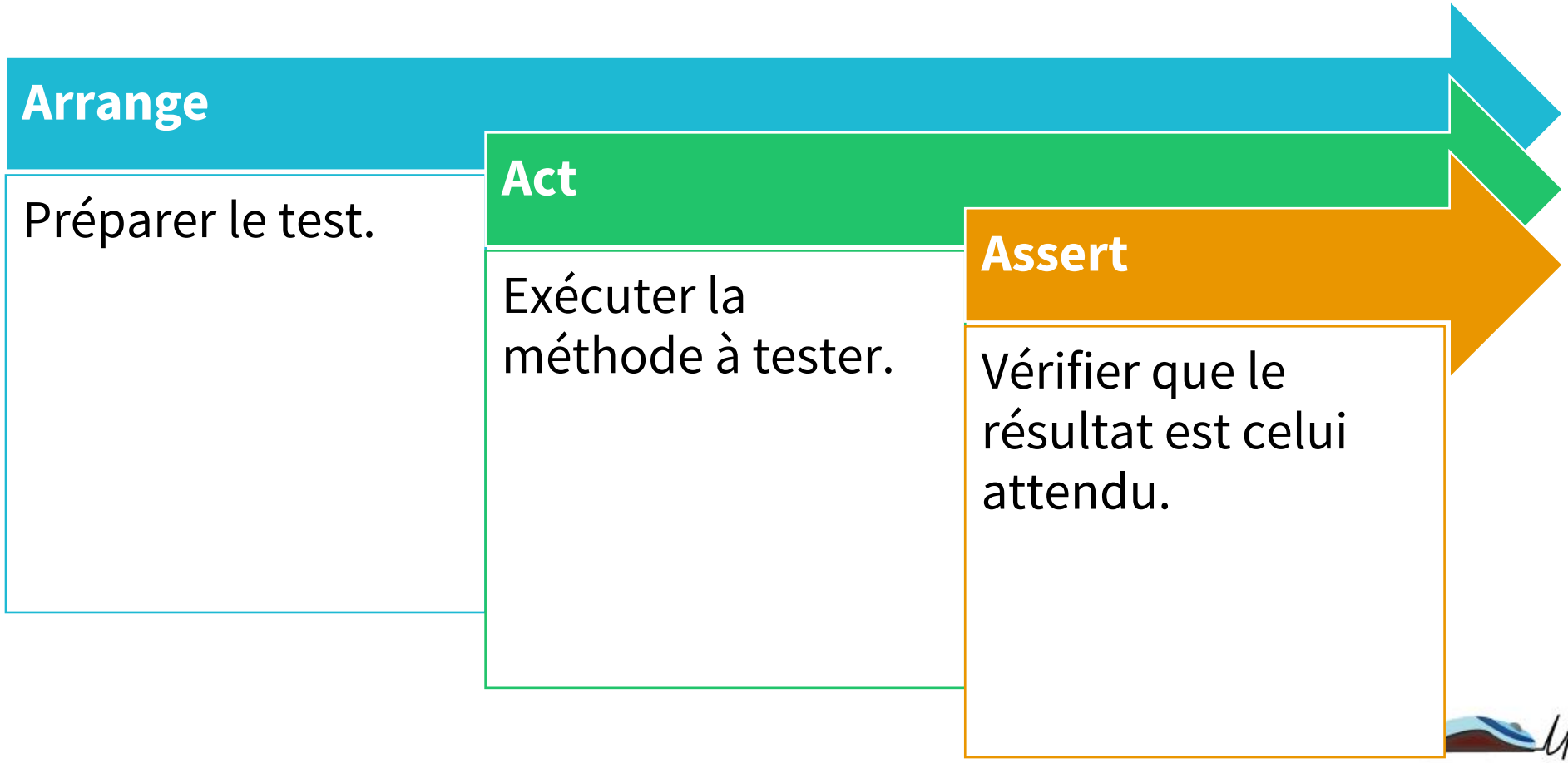
On doit effectuer des tests définis à l'avance

Selon trois grands principes:

- Arrange
- Act
- Assert

Pour définir les tests, on utilise un Framework nommé MSTest

Tests unitaires



Tests unitaires

On va vérifier le comportement de nos méthodes à l'aides des Tests unitaires.

Pour cela on va recréer,

des méthodes spécifiques.

```
[TestClass]
0 références
public class CalculTests
{
    [TestMethod]
    0 références
    public void TestAddition()
    {
        // Arrange
        int a = 5;
        int b = 10;

        // Act
        int result = Calcul.Addition(a, b);

        // Assert
        Assert.AreEqual(15, result);
    }
}
```

Tests unitaires

Les résultats sont affichés de la sorte, grâce à

Exercices pratiques

- Drones