PROGRAMMATION

ORIENTÉ OBJET

IDEE DE BASE

- Toute chose est un objet
- Chaque objet à un type (défini par une Classe)
- Un objet contient des données (Propriétés)
- Tout objet d'un type donné peut comprendre les mêmes messages (Méthodes)
- Un programme est un ensemble d'objets qui communiquent entre eux

OBJET

- Nom
- Propriétés
  - Les données contenues par l'objet (vitesse, couleur, nombre de roues...)
  - types primitifs (int, bool, float, ...)
  - types complexes (objets)
- Méthodes
  - Définissent ce que peut faire l'objet (accélérer, tourner, ...)

INSTANCIATION

- Une classe définit simplement un type d'objet
- On peut créer une nouvelle version d'un objet sur base de cette classe
- La création d'un nouvel objet s'appelle "Instanciation". L'objet ainsi créé est appelé une "Instance"
- Chaque instance a une existance propre dans la mémoire
- La méthode permettant d'instancier un objet s'appelle un "Constructeur"
- Exemples:
  - Voiture verte avec chassis sport
  - Voiture bleue avec chassis SUVdifférentes

ENCAPSULATION

- On "cache" les données à l'intérieur d'un objet
  - propriétés publiques / privées / statiques
- L'objet fournit au monde extérieur les moyens de manipuler ces données (Interface)
  - méthodes publiques / privées / statiques
- Le but est que le monde extérieur n'a pas besoin de comprendre le fonctionnement de l'objet pour le faire fonctionner (exemple: télécommande)

COMPOSITION

- Un objet peut être constitué de plusieurs autres objets (Composants)
- Il fournit au monde extérieur des méthodes permettant de faire fonctionner le tout ensemble.
- Permet de ré-utiliser le même composant dans de nombreux objets différents
- Exemple:
  - Voiture (Roues, Moteur, Chassis)
  - Bâteau (Hélices, Moteurs, Chassis)
  - Caisse à savon (Roues, Chassis)
- Unity est énormément axé sur la notion de composition

HÉRITAGE

- Une classe B peut être définie sur base d'une autre classe A
  - Exemple: Une voiture est une sorte de véhicule
- B hérite de toutes les propriétés et méthodes de A
- B hérite de toutes les propriétés et méthodes dont A hérite
- B définit d'autres propriétés et méthodes qui lui sont propre
- B peut changer l'implémentation des méthodes dont il hérite
- On dit que A est la Super Classe de B
- Une classe ne peut avoir qu'une seule Super Classe

POLYMORPHISME

- Un objet de type B qui hérite d'un type A peut
  - Etre utilisé comme objet de type B
  - Etre utilisé comme objet de type A
- Exemple:
  - "Voiture" et "Bâteau" héritent de "Véhicule"
  - "Véhicule" contient une méthode "Démarrer"
  - "Joueur" peut appeler "Démarrer" sur n'importe quel véhicule

CONVENTIONS C#

- Classe: MaTresChouetteClasse
- Méthode: MaCoolMethode
- Propriété: maProprieteSympathique
- Variables: maBelleVariable
- Bloc de code: "{" et "}"
- Fin de ligne: ";"
- Commentaires: "//" ou "/* */"

SYNTAXE

- Définition d'une classe
  - Class MaTresChouetteClasse:LaSuperClasse {
    //propriétés
    //constructeurs
    //méthodes
    }
- Définition d'un construteur
  - public MaTresChouetteClasse (int monParametre) {
    //code
    }
- Instancier une classe
  - MaTresChouetteClasse monInstance = new MaTresChouetteClasse(3);

SYNTAXE – SUITE

- Définition d'une méthode
  - public void MaCoolMethode(int monParametre) {
    // code
    }
- Appeler une méthode sur une instance
  - monInstance.MaCoolMethode(5);
- Appeler une méthode statique
  - MaTresChouetteClasse.MaMethodeStatique(5);

SYNTAXE – SUITE

- Définir une propriété
  - public int maProprietePublique;
  - public static maProprieteStatique;
  - private bool maProprietePrivee;
- Définir une variable locale
  - int maVariable;
- Accéder à une propriété
  - maVariable += monInstance.maProprietePublique;
  - maVariable += MaTresChouetteClasse.maProprieteStatique;

SYNTAXE – SUITE

- Enchaînement d'appels de méthode
- maVariable = monInstance.MaMethode1().MaMethode2().maPropriete;