Application à la POO – ARIVE – 2

Christophe BLANC

IUT Montluçon
Université Blaise Pascal

Janvier 2016

- ✗ Quid de l'objectif de simuler le positionnement d'un point (cartésien, polaire, latitude, longitude, degré décimal, degré sexagésimal, radian) ?
- ✗ Fastidieux de devoir toujours recréer une nouvelle classe pour affiner un modèle de point.
- ✔ Extraire les attributs communs et les services élémentaires analogues pour les regrouper sous un même nom.
- ✔ Penser en termes de hiérarchie de classes.
- ➠ notions de factorisation, d'héritage (de classe et d'interface) et d'abstraction.

- ✗ Quid de l'objectif de simuler le positionnement d'un point (cartésien, polaire, latitude, longitude, degré décimal, degré sexagésimal, radian) ?
- ✗ Fastidieux de devoir toujours recréer une nouvelle classe pour affiner un modèle de point.
- ✔ Extraire les attributs communs et les services élémentaires analogues pour les regrouper sous un même nom.
- ✔ Penser en termes de hiérarchie de classes.
- ➠ notions de factorisation, d'héritage (de classe et d'interface) et d'abstraction.

- ✗ Quid de l'objectif de simuler le positionnement d'un point (cartésien, polaire, latitude, longitude, degré décimal, degré sexagésimal, radian) ?
- ✗ Fastidieux de devoir toujours recréer une nouvelle classe pour affiner un modèle de point.
- ✔ Extraire les attributs communs et les services élémentaires analogues pour les regrouper sous un même nom.
- ✔ Penser en termes de hiérarchie de classes.
- ➠ notions de factorisation, d'héritage (de classe et d'interface) et d'abstraction.

- ✗ Quid de l'objectif de simuler le positionnement d'un point (cartésien, polaire, latitude, longitude, degré décimal, degré sexagésimal, radian) ?
- ✗ Fastidieux de devoir toujours recréer une nouvelle classe pour affiner un modèle de point.
- ✔ Extraire les attributs communs et les services élémentaires analogues pour les regrouper sous un même nom.
- ✔ Penser en termes de hiérarchie de classes.
- ➠ notions de factorisation, d'héritage (de classe et d'interface) et d'abstraction.

- ✗ Quid de l'objectif de simuler le positionnement d'un point (cartésien, polaire, latitude, longitude, degré décimal, degré sexagésimal, radian) ?
- ✗ Fastidieux de devoir toujours recréer une nouvelle classe pour affiner un modèle de point.
- ✔ Extraire les attributs communs et les services élémentaires analogues pour les regrouper sous un même nom.
- ✔ Penser en termes de hiérarchie de classes.
- ➠ notions de factorisation, d'héritage (de classe et d'interface) et d'abstraction.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

1. **Définir en premier les services que doit réaliser l'objet (fonctionnalités souhaitées)**
   - les définir avec précision ;
   - programmation par "contrat" ;
   - en "Java", utiliser les interfaces.
2. **Choisir ensuite le type (la classe) des attributs**
   - le plus adapté au problème à résoudre ;
   - exploiter les services offerts par les classes des bibliothèques de l'outil de développement ;
   - si remise en cause des choix (des types ou de l'algorithmie), respecter toujours le contrat (possible par l'utilisation des interfaces).
3. **Si besoin de faire évoluer le cahier des charges**
   - modifier le contrat en ajoutant des interfaces ;
   - implémenter ces nouvelles interfaces.

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application
    car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité
    PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie
    d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application
  - car, avec l'expérience, on acquiert :
    - meilleure connaissance des possibilités du langage
    - optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité
    PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

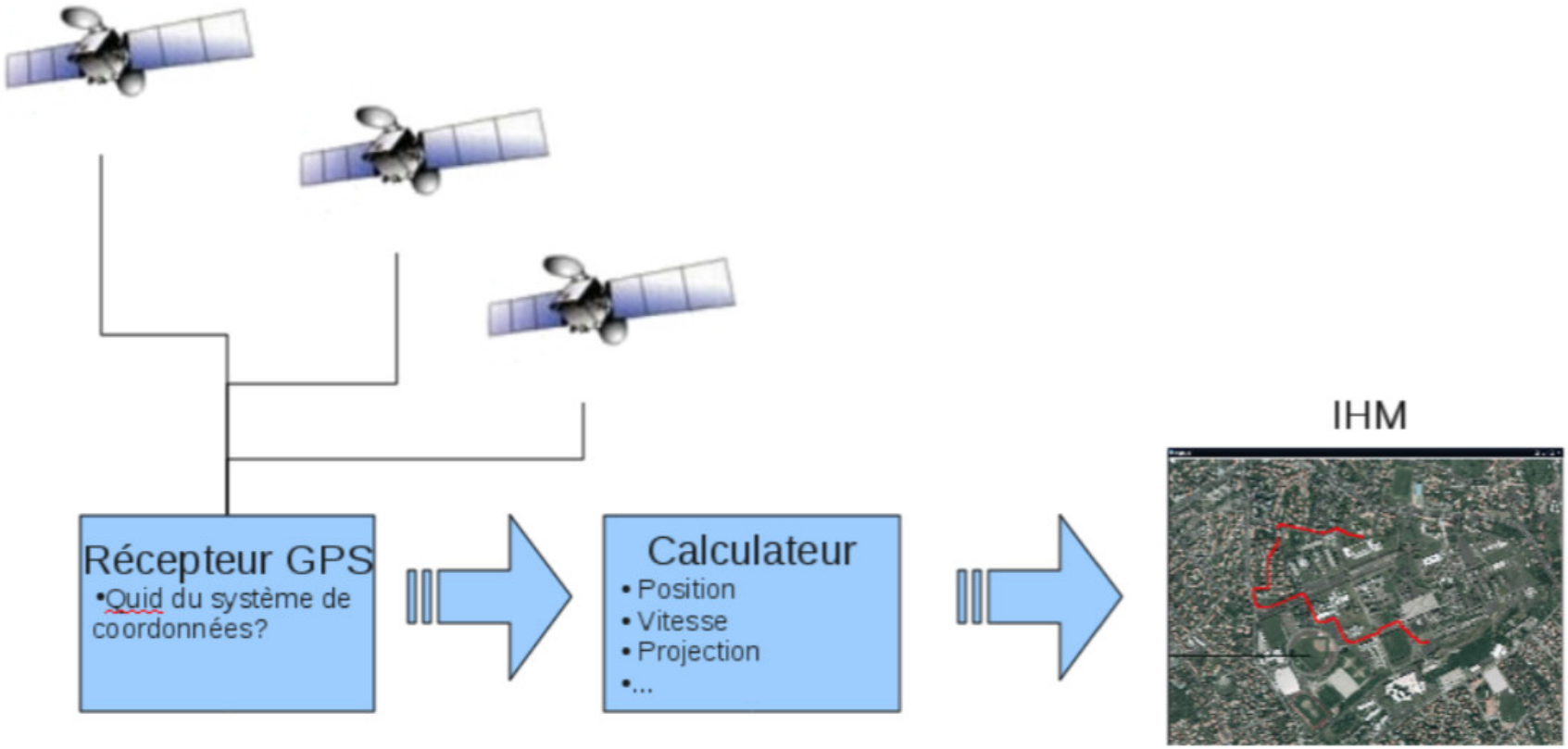
- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité
    PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

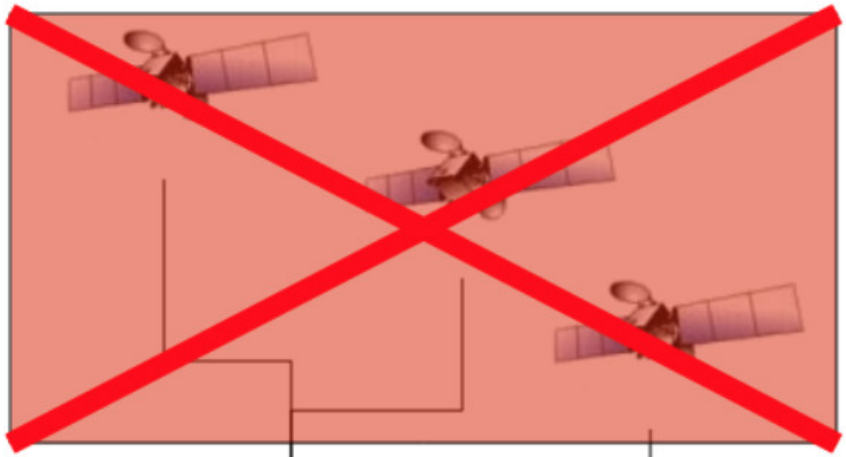
- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité
    PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

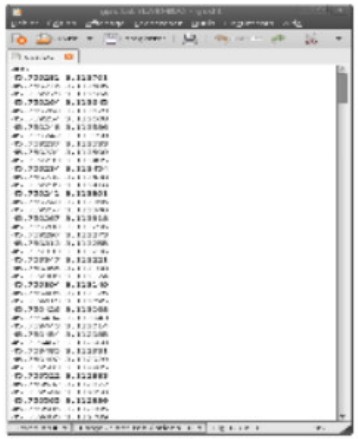
- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

- **Définir le contrat avec les "clients" d'une classe**
  - les services à assurer (notés ISvcXxxx dans nos projets).
- **Retarder le choix du type associé aux attributs (abstraction)**
  - modifier l'implémentation d'une classe sans casser l'application car, avec l'expérience, on acquiert :
    1. meilleure connaissance des possibilités du langage
    2. optimisation des performances (taille, vitesse d'exécution, ...)
- **Mettre en œuvre le polymorphisme pour :**
  - disposer de plusieurs facettes de la même entité PointGPS
  - spécialiser des comportements
    - factorisation
- **Assurer le lien avec :**
  - des objets qui n'appartiennent pas à la même hiérarchie d'héritage
  - le "Runtime - JVM" Java de la machine cible (événements, ...)

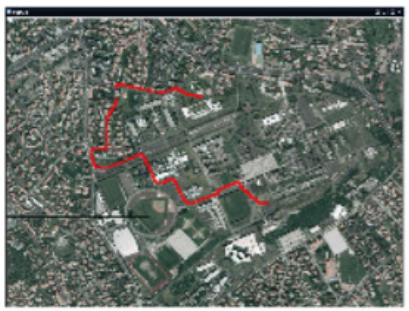

Récepteur GPS
•Quid du système de coordonnées?
Calculateur
• Position
• Vitesse
• Projection
•...
IHM





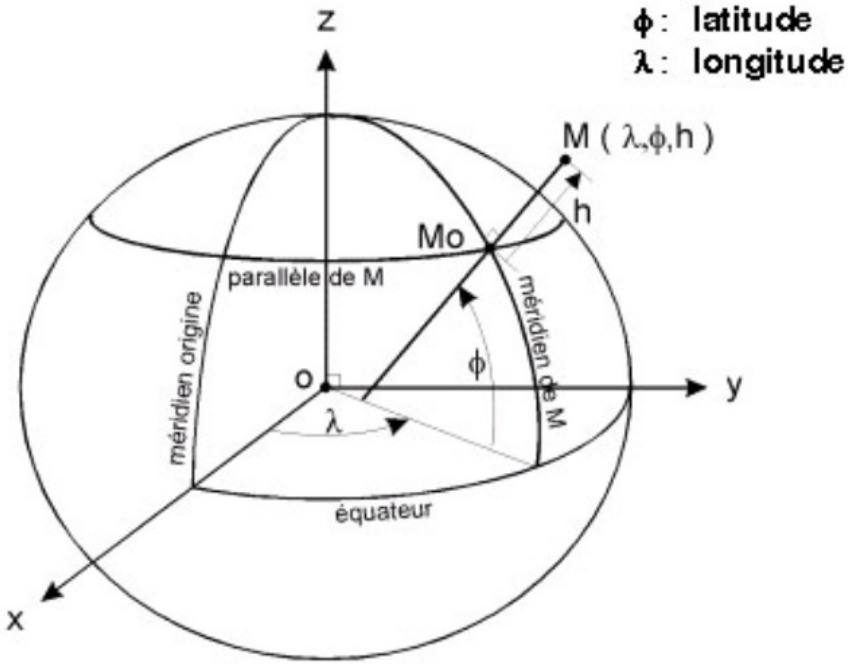
Fichiers de coordonnées
- Lambert 2
- WGS84

IHM



Récepteur GPS
- Quid du système de coordonnées?

Calculateur
- Position
- Vitesse
- Projection
- ...



3 représentations en WGS84 :

- **latitude, longitude en degré décimal : double**
- **latitude, longitude en degré sexagésimal : entier**
- **latitude, longitude en radian : double**

Le contrat :

- stocker une position GPS sous forme latitude, longitude en degré décimal
- connaître la position dans un format "**latitude, longitude en degré décimal**"
- stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- stocker une position GPS sous forme latitude, longitude en radian
- connaître la position dans un format "**latitude, longitude en radian**"

Le contrat :

- stocker une position GPS sous forme latitude, longitude en degré décimal
- connaître la position dans un format "**latitude, longitude en degré décimal**"
- stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- stocker une position GPS sous forme latitude, longitude en radian
- connaître la position dans un format "**latitude, longitude en radian**"

Le contrat :

- stocker une position GPS sous forme latitude, longitude en degré décimal
- connaître la position dans un format "**latitude, longitude en degré décimal**"
- stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- stocker une position GPS sous forme latitude, longitude en radian
- connaître la position dans un format "**latitude, longitude en radian**"

Le contrat :

- stocker une position GPS sous forme latitude, longitude en degré décimal
- connaître la position dans un format "**latitude, longitude en degré décimal**"
- stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- stocker une position GPS sous forme latitude, longitude en radian
- connaître la position dans un format "**latitude, longitude en radian**"

Le contrat :

- stocker une position GPS sous forme latitude, longitude en degré décimal
- connaître la position dans un format "**latitude, longitude en degré décimal**"
- stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- stocker une position GPS sous forme latitude, longitude en radian
- connaître la position dans un format "**latitude, longitude en radian**"

Le contrat :

- ☞ stocker une position GPS sous forme latitude, longitude en degré décimal
- ☞ connaître la position dans un format "**latitude, longitude en degré décimal**"
- ☞ stocker une position GPS sous forme latitude, longitude en degré sexagésimal
- ☞ connaître la position dans un format "**latitude, longitude en degré sexagésimal**"
- ☞ stocker une position GPS sous forme latitude, longitude en radian
- ☞ connaître la position dans un format "**latitude, longitude en radian**"

≪ **ISvcPointGps** ≫
+ *setLatitudeDegDec(latitude : double) : void*
+ *getLatitudeDegDec( ) : double*
+ *setLatitudeDegSexa(degre : int, min : int, sec : int) : void*
+ *getLatitudeDegSexa( ) : int[]*
+ *setLatitudeRad(latitude : double) : void*
+ *getLatitudeRad( ) : double*
+ *setLongitudeDegDec(longitude : double) : void*
+ *getLongitudeDegDec( ) : double*
+ *setLongitudeDegSexa(degre : int, min : int, sec : int) : void*
+ *getLongitudeDegSexa( ) : int[]*
+ *setLongitudeRad(longitude : double) : void*
+ *getLongitudeRad( ) : double*

rmq : abstrait (abstract, noté en italique dans UML)

```
public interface ISvcPointGPS
{

   public abstract void setLatitudeDegDec(double latitude);

   public abstract double getLatitudeDegDec();

   public abstract void setLongitudeDegDec(double longitude);

   public abstract double getLongitudeDegDec();

   public abstract void setLatitudeRad(double latitude);

   public abstract double getLatitudeRad();

   public abstract void setLongitudeRag(double longitude);

   public abstract double getLongitudeRad();

   public abstract void setLatitudeDegSexa(int degre, int min, int sec);

   public abstract int[] getLatitudeDegSexa();

   public abstract void setLongitudeDegSexa(int degre, int min, int sec);

   public abstract int[] getLongitudeDegSexa();
}
```

Exercice 2
Enoncé

1. Créer un nouveau projet BlueJ et saisir l'interface *ISvcPointGPS*.
2. Créer 3 classes *PointDegDec*, *PointDegSexa*, et *PointDegRad* qui implémentent (toutes les méthodes) de l'interface *ISvcPointGPS*.
3. Effectuer les tests pour vous assurer du codage correct des classes.