Bases de la POO / Java

Quelques classes standard
Math, System, Object,...

Package java.lang

- **Ensemble de classes prédéfinies qui proposent des services**

- **Principales classes (importation implicite)**
  - La classe **Math**
  - La classe **System**
  - La classe **Object**
  - Les classes enveloppes (wrappers)
  - La classe **Class**
  - La classe **RunTime**
  - …

- http://docs.oracle.com/javase/7/docs/api/java/lang/package-summary.html

La classe Math

- **Principales fonctions mathématiques**
  - Constantes E, PI
  - Fonctions ***min***, ***max***, ***round***, …
  - Fonctions ***abs***, ***pow***, ***sqrtr***, …
  - Fonctions trigonométriques (***sin***, ***cos***, ***tan***, …)
  - Fonctions ***exp*** et ***log***
  - Génération de nombres aléatoires (fonction ***random***)

- http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

La classe System

- La classe System centralise l'accès :
  - aux trois flux de base : in, out et err
  - à l'horloge du système d'exploitation
  - aux fonctions utilitaires de la JVM

- http://docs.oracle.com/javase/7/docs/api/java/lang/System.html

La classe System

- **Principales méthodes de classe**
  - Méthodes ***currentTimeMillis, nanoTime***
  - Méthode ***exit***
  - Méthodes ***getProperty*** et ***setProperty***
  - Méthode ***gc***
  - …

- **Attributs de classe associés aux flux standards**
  - **System.*out*** Classe **PrintStream**
  - **System.*in*** Classe **PrintStream**
  - **System.*err*** Classe **PrintStream**

System.out.***println***(---);
System.out.***printf***("format-string" [, arg1, arg2, …]);
System.exit (0);→**Forcer la fin d'exécution du programme**

Hiérarchie java.lang

- Les deux classes Math et System font partie du package lang, importé par défaut.

- Toutes les classes du package sont organisées en hiérarchie.
- Elles sont toutes dérivées de la classe **Object**, base de la hiéarchie.

- https://docs.oracle.com/javase/7/docs/api/java/lang/package-tree.html

La classe Object (1)

- Cette classe contient (sous forme de méthodes) les servitudes de base pour la gestion des objets.

- **Transmet implicitement toutes ses méthodes à toute classe Java**
  - Relation d'héritage (Cf cours suivant)
  - Induit la nécessité de **redéfinir** ces méthodes dans toute classe Java
  - Transtypage implicite possible de toute référence sur un objet d'une classe quelconque dans une variable de type **Object** (analogie avec type void* du langage C)

La classe Object (2)

- **Met à disposition un constructeur par défaut**

- Principales méthodes (**d'instance**)
  - toString : retourne un descriptif de l'objet cible
  - equals : prédicat d'égalité de deux objets
  - clone : retourne un duplicata de l'objet cible
  - getClass : retourne la classe de l'objet cible
  - …

Comparaison d'objets (1)

- **Identité de deux objets**
  - Nécessité de la même classe génitrice
  - Deux objets sont identiques s'ils possèdent la même référence Java
  - Opérateur == pour les comparer
- Exemple :

```
Point A= new Point (1., -2.5);
Point B= A;
if (A == B) Vrai
```

Comparaison d'objets (2)

- **Egalité de deux objets**
  - Nécessité de la même classe génitrice
  - Deux objets sont égaux s'ils possèdent les mêmes valeurs d'attributs
  - Tout ou partie des attributs (au gré du programmeur)
- Exemple :

```
Point A= new Point (1., -2.5);
Point B= new Point (1., -2.5);
if (A == B) Faux
```

Comparaison d'objets (3)

- **Egalité de deux objets**
  - Méthode ***equals*** obligatoire pour les comparer
  - Redéfinition obligatoire de cette méthode dans la classe génitrice

- Signature formelle imposée pour cette méthode

  public boolean equals (Object op2)

- Exemple :

  Point A= new Point (1., -2.5);
  Point B= new Point (1., -2.5);
  if (A.equals(B)) on veut obtenir Vrai

Classe Point – méthode equals


```
public boolean equals (Object op2)
{
   double deltaX= abscisse – ((Point)op2).abscisse;
   double deltaY= ordonnee - ((Point)op2).ordonnee;
   return (Math.abs(deltaX)<= EPSILON &&
           Math.abs(deltaY)<= EPSILON);
}

public boolean equals (Object op2)
{
   return this.confondus(((Point)op2));
}
```

Duplication d'objets (1)

- **Deux possibilités complémentaires**
  - Définition d'un constructeur de copie
  - Redéfinition de la méthode ***clone***

- Signature formelle imposée pour cette méthode

  public Object clone ()

Duplication d'objets (2)

- **Constructeur de copie de la classe Point**

```
public Point (Point P)
{
    abscisse = P.abscisse;
    ordonnee = P.ordonnee;
}
```

- Mode de duplication analogue au langage C/C++

Duplication d'objets (3)

- **Redéfinition de la méthode clone**

```
public Object clone ()
{
  return new Point (abscisse, ordonnee);
}
```

- Mode de duplication introduit par le langage Java