

BASES DE LA PROGRAMMATION OBJET

Martine GAUTIER - Université de Lorraine
martine.gautier@univ-lorraine.fr

Objectifs

- Apprendre les constructions spécifiques des langages de programmation objet
 - objet, classe, héritage, polymorphisme, liaison dynamique, exceptions
 - Java
- Se familiariser avec la conception de programmes objets
 - langage de modélisation UML
- Acquérir une bonne méthode de développement



Enseignement sur tout le semestre

16h CM 16h TD 28h TP

Modalités de contrôle des connaissances

TPs à déposer chaque semaine, questionnaires, devoir, écrit à la fin

$$\text{Note UE} = 0,2 * E1 + 0,2 * E2 + 0,4 * E3$$

Épreuve de seconde chance

$$\text{Note finale} = \text{Max} (\text{Note1}, 0.5*\text{Note1}+0.5*\text{Note2})$$

INTRODUCTION

Martine GAUTIER - Université de Lorraine
martine.gautier@univ-lorraine.fr

❏ Programmer

- ❏ écrire un texte dans un langage compréhensible par la machine

❏ Programmation impérative

- ❏ le texte est une suite d'instructions, exécutées l'une après l'autre



Données

Instructions

❏ Programmer

- ❏ écrire un texte dans un langage compréhensible par la machine

❏ Programmation impérative

- ❏ le texte est une suite d'instructions, exécutées l'une après l'autre



Données

Nombre, chaîne de caractères, tableau, liste, *etc.*

Instructions

Affectation, condition, itération

❏ Programmer (sans objet)

- ❏ La notion de fonction est prépondérante.
- ❏ Une fonction calcule un résultat à partir de données passées en paramètre.

= première forme de programmation (1950) directement issue des maths



Données

Instructions

❏ Programmer avec des objets

- ❏ La notion d'objet est prépondérante.
- ❏ On crée des objets sur lesquels on applique des traitements.



❏ Programmer avec des objets

- ❏ La notion d'objet est prépondérante.
- ❏ On crée des objets sur lesquels on applique des traitements.
- ❏ Paradigme de programmation issu du monde réel
 - ❏ téléphone, voiture, calculatrice, chronomètre, etc.
- ❏ Apparue dans les années 1970
- ❏ À long terme, les objets sont plus pérennes que les fonctions.

❏ Langages objets

- ❏ 1972 Smalltalk, ancêtre des LO
- ❏ Eiffel, C++, C#, Java, Python, PHP, Ruby, Dart, *etc.*
- ❏ Java : essor lié à Internet
- ❏ les paradigmes de PO se retrouvent dans tous les LO
 - ❏ objet, classe, héritage, liaison dynamique

❏ Bibliographie

- ❏ pléthore de livres, de sites Internet
- ❏ Java 11
- ❏ <https://openclassrooms.com/courses/apprenez-a-programmer-en-java>
- ❏ <http://java.developpez.com/cours>
- ❏ <https://docs.oracle.com/javase/tutorial>

PROGRAMMER avec des OBJETS

Martine GAUTIER - Université de Lorraine
martine.gautier@univ-lorraine.fr

Classe et objet

- ❑ Une chaîne de fabrication fabrique des objets identiques.
 - ❑ ex : téléphone, voiture, etc.
- ❑ A l'achat, rien ne différencie deux objets fabriqués issus de la même chaîne de fabrication
 - ❑ ils disposent de données propres qui évoluent au cours de leur vie.
- ❑ Classe = chaîne de fabrication d'un objet

Programme objet

- ❑ = suite d'instructions exécutées séquentiellement
- ❑ Instructions
 - ❑ créer un objet à partir d'une classe
 - ❑ utiliser un objet
- ❑ Pour débiter, on travaillera avec des classes toutes faites, avant d'apprendre à en faire des nouvelles.

- ❑ Les instructions sont spécifiques à chaque classe.
 - ❑ on ne crée pas un téléphone de la même façon qu'une voiture
 - ❑ on n'utilise pas un agenda de la même façon qu'une tondeuse.
- ❑ Tous les objets instances d'une même classe se créent et s'utilisent de la même façon.
- ❑ Pour chaque classe :
 - ❑ comment créer un objet ?
 - ❑ que peut-on en faire ?



Documentation d'une classe Java

- ❑ = description précise des méthodes de création et d'utilisation des objets
- ❑ Indispensable pour pouvoir utiliser la classe
- ❑ Page HTML
 - ❑ ex : la classe Chronometre

chronometre

Class Chronometre

java.lang.Object
chronometre.Chronometre

Nom de package

Nom de classe

```
public class Chronometre  
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Chronometre()

Method Summary

Methods

Modifier and Type

Method and Description

void

arreter()

void

demarrer()

int

getTemps()

void

raz()

void

suspendre()

- Une classe est rangée dans un package, dont le nom commence par une minuscule.
- Le nom de la classe commence par une majuscule ; c'est un identifiant unique dans le package.
- Le nom de classe complètement qualifié inclut le package ; c'est un identifiant unique.
- Les constructeurs sont utilisés pour fabriquer un objet (instanciation). Ils portent le nom de la classe.
- Les fonctions d'observation sont utilisées pour observer une caractéristique de l'objet.
- Les fonctions de transformation sont utilisées pour modifier l'état de l'objet.

Premier programme Java



Apprendre à écrire correctement deux sortes d'instructions, en lisant la documentation de la classe

- ❑ création d'un objet
- ❑ utilisation d'un objet

Création / Instanciation

```
Chronometre c = new Chronometre ( ) ;
```



Création / Instanciation

Chronometre

c

variable destinée à contenir
l'adresse de l'objet, pour
pouvoir s'en servir plus tard

=

new

Chronometre ()

;

mot-clé

utilisation du
constructeur,
conforme à la doc

déclaration unique de la variable
c qui ne pourra pas contenir
autre chose qu'une adresse de
Chronometre

adresse mémoire de l'objet



Utilisation/Appel de fonction

Uniquement les fonctions citées dans la doc. en respectant le profil

```
c.demarrer ( ) ;
```



Utilisation/Appel de fonction

Uniquement les fonctions citées dans la doc. en respectant le profil

`c.demarrer () ;`

receveur
= objet à qui on s'adresse

appel d'une fonction de transformation
conforme à la doc.

En C, on écrirait : `demarrer(c)`



Utilisation/Appel de fonction

Uniquement les fonctions citées dans la doc. en respectant le profil

```
c.demarrer ( ) ;
```

```
c.suspendre() ;
```

```
c.arreter() ;
```

```
c.raz() ;
```

```
int t = c.getTemps() ;
```

// Attention, t contient un entier, pas une adresse



Un programme avec plusieurs objets

```
Chronometre c1 = new Chronometre() ;
```

```
c1.demarrer() ;
```

```
Chronometre c2 = new Chronometre() ;
```

```
c2.demarrer() ;
```

```
c1.suspendre() ;
```

```
int t1 = c1.getTemps() ;
```

```
c1.demarrer() ;
```

```
c2.raz() ;
```



Réutiliser les variables

```
Chronometre c1 = new Chronometre() ;
```

```
c1.demarrer() ;
```

```
c1 = new Chronometre () ;
```

```
c1.demarrer() ;
```

```
Chronometre c2 = new Chronometre() ;
```

```
t1 = c1.getTemps() ;
```

```
c2 = c1 ; // copie d'adresse
```

```
int t2 = t1 ; // copie d'entier
```



Un autre exemple : la classe `date.Date`

- Etude de la documentation
- Quoi de neuf ?
 - plusieurs constructeurs avec des profils différents
 - types `boolean`, `String`
- Un exemple de programme utilisant des instances de `Date`



Method Summary

Methods

Modifier and Type	Method and Description
int	annee () Année de la date
boolean	anneeBissextile () S'agit-il d'une année bissextile ?
void	avancer (int nbJours) Avancer d'un certain nombre de jours
int	jour () Jour de la date
int	mois () Mois de la date
void	reculer (int nbJours) Reculer d'un certain nombre de jours
void	setAnnee (int annee) Fixer l'année
void	setJour (int jour) Fixer le jour
void	setMois (int mois) Fixer le mois
java.lang.String	toString () Sous la forme jour/mois/année

Un autre exemple : la classe Point

- Etude de la documentation
- Quoi de neuf ?
 - type double
- Attention au profil de la fonction distance
- Un exemple de programme utilisant des instances de Point



Class Point

Constructor Summary

Constructors
Constructor and Description
<code>Point(double a, double o)</code> initialiser un point a partir de ses coordonnees cartesiennes
<code>Point(Point p)</code> Constructeur de copie

Method Summary

Methods	
Modifier and Type	Method and Description
void	deplacer (double ca, double co) Translater le point
double	distance (Point p)
double	getAbscisse ()
static int	getCount () Nombre d'instanciations de points
double	getOrdonnee ()
double	rho ()
double	teta ()
java.lang.String	toString ()