Cours 2.2

I. Composition

I.1 Exemple du véhicule : x, y, chassis, roueAv, roueAr (voir dessin1 et dessin2) §
I.2 Relation entre objets ==> diagramme d'objets [**] (voir dessin) §
I.3 ==> diagramme de classes [**] (voir dessin) §
I.4 Recopie, clonage simple (voir dessins) §
I.5 Appel de constructeurs dans un constructeur :
un constructeur peut en appeler un autre sur le même objet par `this( paramètres );`
en première instruction du constructeur
[**] Télécharger le viewer PowerPoint si nécessaire.

II. Méthodes particulières

II.1 Accesseurs (ou fonctions d'accès ou getters) :

```
public typeAttribut getNomAttribut()
  { return this.aNomAttribut; }
```

protection ==> ne peut être modifié
II.2 Modificateurs (ou procédures de modification ou setters) :

```
public void setNomAttribut( typeAttribut pNomParamètre )
  { this.aNomAttribut= pNomParamètre; }
```

protection ==> tests de cohérence, maintenabilité
II.3 Fonctions d'état (accesseurs particuliers, booléens) :

```
public boolean isAdjective()
  { ... return valeurBooléenne; }
```

III. Expressions

III.1 Expression référence
1.1) valeur littérale : `null` et `this`
1.2) variable : `vC1` ou `pV1` ou `aV1`
1.3) `new NomClasse()`
1.4) appel de fonction : getRoueAv()
III.2 Expression arithmétique (entière)
2.1) valeur littérale : `-123` ou `+45` ou `67`
2.2) variable : (locale, attribut, paramètre)
2.3) constante : (locale, attribut, paramètre) déclarée comme suit :
- `final int CONSTANTE_LOCALE = 10;`
- `private final int ATTRIBUT_CONSTANT = 10;`
- `final int pNomParametre`

==> reste constant pendant la méthode
2.4) appel de fonction : getDiameter()
2.5) *OP1*(*expression_arithmétique*) : - ou +
2.6) expression_arithmétique *OP2* expression_arithmétique : `+ - * / %`
[division entière : `/` → quotient, `%` → reste]
III.3 Expression booléenne
3.1) valeur littérale : `true` ou `false`
3.2) variable : (locale, attribut, paramètre)
3.3) constante : (locale, attribut, paramètre) déclarée comme suit :
- locale : inutile

- `private final boolean ATTRIBUT_CONSTANT = true;` : rarement utile
- `final boolean pNomParametre`

==> reste constant pendant la méthode

3.4) appel de fonction : isVisible()

3.5) résultat d'une comparaison avec : `< > <= >= == !=`

3.6) *OP1*(*expression_booléenne*) : `!` → NON

3.7) expression_booléenne *OP2* expression_booléenne : `&&` → ET ou `||` → OU

3.8) évaluation paresseuse : s'arrête dès qu'on peut déterminer le résultat (faux avec ET, vrai avec OU) ==> non commutatif (mais même résultat)

III.4 Priorité des opérateurs : voir ce tableau

IV. Tests

IV.1 SI ALORS : `if ( expression booléenne ) { instructions si vraie }`

parenthèses obligatoires même si simplement `(ok)`, certaines instructions écrites ne sont jamais exécutées

IV.2 SI ALORS SINON : `if ( expression booléenne ) { instructions si vraie } else { instructions si fausse }`

IV.3 une instruction sans `{}` ==> danger ! (en cas d'ajout ou d'imbrication)

IV.4 if successifs non indépendants

```
if (a<0) S.o.p("neg");   if (a>0) S.o.p("pos");   if (a==0) S.o.p("nul");
```

- amélioration 1 : ajouter 2 `else` - amélioration 2 : supprimer le 3ème `if`

V. Récursivité

V.1 Une définition : découper un problème en une multitude de sous-problèmes similaires plus simples.

Souvent une manière très naturelle de résoudre un problème complexe. Exemple des tours de Hanoï .

V.2 Exemple : On suppose savoir faire {AfficherUne *}

{AfficherLes N *} = {AfficherUne *} PUIS {AfficherLes N-1 *}

et ainsi de suite jusqu'au cas terminal : N vaut 1. Dans ce cas, inutile de rappeler {AfficherLes 0 *} !

V.3 En java, une méthode peut s'appeler elle-même sans aucune déclaration particulière.

V.4 Élégant mais pas le plus efficace ni en temps ni en mémoire. Répéter des instructions efficacement ==> boucle *(voir cours suivant)*

V.5 Exercice : écrire la fonction factorielle en utilisant la récursivité.

Aide : règle de récurrence $n! = n \times (n-1)!$ et cas terminal $n \leq 1 \rightarrow 1$

Lire le poly :

tout jusqu'à la section 2.2, sections **3.1**, 4, 5.2.0, **6**, 7.1, 7.2, 8.1, **8.2.1.1**, et annexes 6 & 7