Java cours 3

**Classes et objets**

**Les Classes :**

**Diapo 2**

Sémantique commune = comportement et structure identique

Un objet créé à partir d’une classe, il est lié à cette classe au cours de toute son existence. Il n’y a qu’une seule classe par objet (pas d’héritage multiple).

**Diapo 4**

Le nom de la classe commence toujours par une Maj, nom de méthode commence toujours par une minuscule.

**Diapo 5**

Les variables déclarées dans la classe sont globales à cette classe. Elles peuvent être utilisées par toutes les méthodes de la classe.

**Diapo 6**

Attention les types de bases ne sont pas des objets.

**Diapo 7**

Une méthode a toujours un type de retour. Ce type peut être void, un type de base ou un type construit. Pour nommer une fonction on utilise le camelcase, càd la première lettre de la méthode est en minuscule les autres mots composant le nom commencent par une maj.

On peut renvoyer une simple valeur, un objet, ou même un tableau d’entiers (int[]).

Liste des paramètres, de zéro au nombre nécessaire pour la méthode donnée.

On peut avoir plusieurs return dans une méthode. Mais attention si méthode avec beaucoup de lignes mettre un return en plein milieu cela peut être dangereux. Pour une méthode courte cela peut être très efficace.

Une variable déclarée dans une méthode est locale à cette méthode.

**Diapo 15**

Une variable est visible dans le bloc défini par les {…} dont elle est entourée. Une variable peut être définie n’importe où dans un bloc, mais elle est connue depuis son point de déclaration jusqu’à la fin du bloc. Cela permet d’avoir une meilleure localité des variables, c’est-à-dire proche de l’endroit où on va les utiliser.

**Les Objets :**

Une instance d’une classe et d’une seule.

La référence est le nom que l’on donne à l’objet. Le new est l’équivalent du malloc en C.

Passage des paramètres est un passage par valeur. Bien revoir cette diapo. Il faut bien se rappeler qu’une référence d’un objet c’est un pointeur.

L’objet courant « this » permet de lever des ambigüités. Bien revoir la diapo du this avec Dessin.

**Encapsulation :**

Permet de protéger les variables et méthodes de la classe que l’on a créée.

On ne peut pas faire p.x si l’attribut x de la classe est définit par la visibilité private. Par contre on peut ajouter des méthodes get et set (getter / setter) pour consulter et modifier ces attributs. Dans NetBeans on peut faire un clic droit et ajouter code pour générer directement les méthodes get et set des variables private. Si on veut éviter que la personne rentre n’importe quelle valeur on peut mettre des conditions dans le set pour éviter les valeurs incorrectes.

L’encapsulation permet de contrôler l’utilisation des objets que l’on a définis, c’est la robustesse. Autre point positif c’est la facilité d’implémentation du code. Le code d’une méthode peut être complétement changé sans que ça n’impacte l’utilisateur, si la signature est toujours la même.

**Objet String :**

Attention l’objet String a un comportement particulier.

Si on fait :

String s1 = « toto »

String s2 = « toto »

Le compilateur voit que c’est les deux mêmes objets et fait donc pointer s2 sur s1 et ne créé pas un objet propre à s2. Si s2 avait été créé avec le mot new alors on aurait deux objets différents ayant le même contenu.