

BPO

TP 2 - Iceberg2D

Exercice 1 - Quelques schémas mémoire

```
package japan ;

public class Ysuka {

    private char c ;
    private Maya m ;
    private Tsuki t ;

    public Ysuka (char c, Maya m, Tsuki t) {
        this.c = c ;
        this.m = m ;
        this.t = t ;
    }

    public Ysuka (Maya m) {
        c = 'c' ;
        this.m = m ;
        t = new Tsuki(m.getVal()) ;
    }

    public Ysuka (Tsuki t) {
        this.t = t ;
        this.c = '.' ;
        this.m = null ;
    }

    public Ysuka (Ysuka y) {
        c = y.getC() ;
        if (y.getM() != null) {
            m = new Maya(y.getM()) ;
        }
        if (y.getT() != null) {
            t = new Tsuki(y.getT()) ;
        }
    }

    public char getC () {
        return c ;
    }

    public Maya getM () {
        return m ;
    }

    public Tsuki getT () {
        return t ;
    }
}
```

```
package japan ;

public class Maya {

    private int val ;

    public Maya (int v) {
        val = v ;
    }

    public Maya (Maya m) {
        val = m.getVal() ;
    }

    public int getVal () {
        return(val) ;
    }
}

package japan ;

public class Tsuki {

    private int val ;
    private Ysuka y ;

    public Tsuki (int v) {
        val = v ;
        y = null ;
    }

    public Tsuki (int v, Ysuka y) {
        val = v ;
        this.y = y ;
    }

    public Tsuki (Tsuki t) {
        val = t.getVal() ;
        if (t.getY() != null) {
            y = new Ysuka(t.getY()) ;
        }
    }

    public int getVal () {
        return val ;
    }

    public void setY (Ysuka y) {
        this.y = y ;
    }


    public Ysuka getY() {
        return y ;
    }
}
```

À l'aide du code des 3 classes **Ysuka**, **Tsuki** et **Maya** du package **japan** donné ci-dessus, dessinez, sur une feuille de papier, les schémas mémoire correspondant aux deux séquences de code suivantes :

| code0 | code1 |
|--|--|
| <pre>Maya m1 = new Maya(5) ; Maya m2 = m1 ; Maya m3 = new Maya(m1) ;</pre> | <pre>Maya m1 = new Maya(5) ; Ysuka y1 = new Ysuka('y', m1, new Tsuki(10)) ; Tsuki t ; Maya m2 = new Maya(1) ; Ysuka y2 = new Ysuka(m2) ;</pre> |


Exercice 2 - Le logiciel artEoz


1. Depuis l'url **artez.loria.fr**, cliquez sur le lien “**accès à la version en ligne du logiciel artEoz**”.
2. À gauche, dans la fenêtre d'édition de code, copiez-collez **code1** donné à l'exercice précédent, à partir du fichier donné sur arche (le copier-coller à partir du fichier pdf de cet énoncé ne fonctionne pas avec les symboles).
Le code entré correspond aux instructions de la fonction **main** d'une classe de test.

3. Cliquez sur le bouton  pour afficher le schéma mémoire correspondant à l'exécution de toutes les lignes du code donné.

Le schéma affiché correspond-il à votre dessin fait à la main ?

Cherchez les différences et essayez de comprendre. Aidez-vous des boutons qui permettent d'exécuter les instructions

une par une (ce que l'on appelle le mode “pas à pas”) : 

4. On rappelle qu'un objet mort est une instance de classe qui n'est plus référencée par une variable ou un champ de classe. Ajoutez une ou plusieurs instructions dans le code afin de provoquer la présence d'un ou plusieurs objets morts et vérifiez leur présence avec l'option **Objets morts** sélectionné dans le menu ouvert par  ; cette option permet de visualiser en grisé les instances des classes qui ont été construites, mais qui ne sont plus accessibles.

5. Remplacez le code par celui-ci :

| code2 |
|--|
| <pre>Maya ma = new Maya(6) ; for (int i = 0 ; i < 4 ; i++) { Maya m = new Maya(i) ; int k = m.getVal() ; }</pre> |

La variable **i**, indice de la boucle, est-elle encore accessible (utilisable) après l'itération ? Même question pour la variable **m**. Pourquoi ?

6. Combien y-a-t'il d'objets morts construits après l'exécution de ce code ? Vérifiez avec **artEoz**.
7. Remplacez le code par celui-ci :

| code3 |
|--|
| <pre>Maya m = new Maya(6) ; int k ; k = m.getVal() ;</pre> |

Sélectionnez l'option **Pile** du menu des options  . Exécutez le code en mode “pas à pas”.
Que pouvez-vous ainsi visualiser ?

- À la main, dessinez le schéma mémoire correspondant au code suivant. Vous pouvez vérifier simultanément votre schéma avec celui produit par artEoz, en mode pas à pas.

code4

```
Maya m1 = new Maya(7) ;
Maya m2 = m1 ;
Maya m3 = new Maya(m1) ;

Tsuki t1 = new Tsuki(0) ;
Ysuka y1 = new Ysuka('f', m3, new Tsuki(45))
;
t1.setY(y1) ;

Tsuki t2 =new Tsuki(t1) ;
```

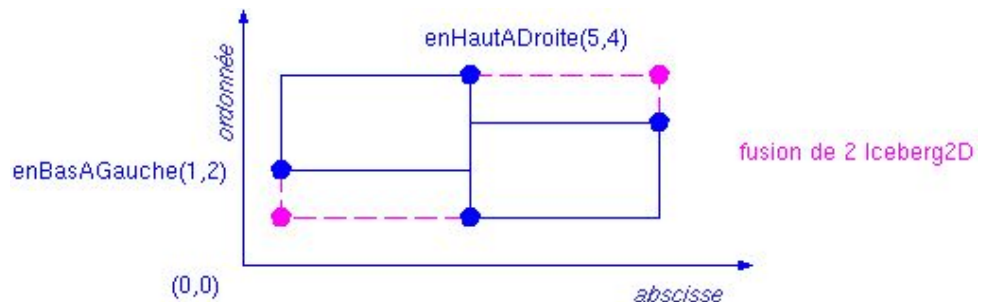
Exercice 3 - La classe Iceberg2D


- Copiez chez vous, dans le bon répertoire, la classe **Iceberg2D** dont le code (incomplet) est donné sur arche. Pour trouver le nom du répertoire dans lequel stocker le fichier, regardez le contenu du fichier...
- Générez la documentation avec la commande **javadoc** :

javadoc -classpath /opt/depot/BPO/geometrie.jar:. -encoding "utf-8" -charset "utf-8" -d glaces/javadoc glaces

Vous pouvez ensuite consulter cette documentation avec votre navigateur en chargeant le fichier **index.html**.

- À l'aide de cette documentation, complétez progressivement le code des fonctions de la classe **Iceberg2D** y compris la fonction **main** permettant de tester votre code. Chaque fonction est supposée travailler sur des paramètres corrects. Testez les fonctions écrites au fur et à mesure que vous les écrivez.



- À la fin de la séance, déposez le fichier **Iceberg2D.java** sur arche. Si nécessaire, terminez la programmation avant la prochaine séance de TP et déposez le code sur arche.
- Utilisez le logiciel **artEoz** pour tester votre application **Iceberg2D** :
 - Chargez le fichier **Iceberg2D.class** à l'aide du bouton 
 - copier/coller une partie du corps de la fonction main de votre classe **Iceberg2D** dans la fenêtre de code. Attention, il faut, avec **artEoz**, que les noms de variables commencent obligatoirement par une lettre minuscule.