Cours du 19 octobre

E) Constructeurs et héritage

- Le constructeurs ne sont pas des méthodes comme les autres:
  - le redéfinition n'a pas de sens.
- Appeler un constructeur dans un constructeur:
  - super() appelle le constructeur de la super classe
  - this() appelle le constructeur de la classe elle-même
  - Ces appels doivent se faire au début du code du constructeur

Constructeurs

- Principes:
  - Quand une méthode d'instance est appelée l'objet est déjà créé.
  - Création de l'objet (récursivement)
    1. Invocation du constructeur de la super classe
    2. Initialisations des champs par les initialisateurs et les blocs d'initialisation
    3. Une fois toutes ces initialisations faites, appel du corps du constructeur (super() et this() ne font pas partie du corps)

Exemple

```
X
            M    =0 00   ;
               M    ;
      X()
       M     =   M    ;

            (          )
      (     &      M    );


         X
            M     = 0    00;
       ()
       M      =   M    ;
```

Résultat

	xMask	yMask	fullMask
Val. par défaut des champs	0	0	0
Appel Constructeur pour Y	0	0	0
Appel Constructeur pour X	0	0	0
Initialisation champ X	0x00ff	0	0
Constructeur X	0x00FF	0	0x00FF
Initialisation champs de Y	0x00FF	0xFF00	0x00FF
Constructeur Y	0x00FF	0xFF00	0xFFFF

La classe Object

- Toutes les classes héritent de la classe Object
- méthodes:
  - public final Class<? extends Object> **getClass**()
  - public int **hashCode**()
  - public boolean **equals**(Object obj)
  - protected Object **clone**() throws CloneNotSupportedException
  - public String **toString**()
  - protected void **finalize**() throws Throwable
  - (wait, notify,notfyall)

Exemple

```
   A
       ;
       ;
A(      ,      )
         . = ;     . = ;

   D <T>
T  ;
D(T  )
         . = ;
```

Suite

```
                    (S          )
A  =    A(1,2);
A  =    A(1,2);
A  = ;
   ( == )
    S     .   .       (" == ");

    S     .   .       (" != ");
   ( .      ( ))
    S     .   .       ("         ");

     S     .   .       ("             ");
S     .   .       ("O      : "+ .  S     ()+"       "+ .   C    ());
S     .   .       (" .    C   ()"+ .    C   ());
S     .   .       (" .    C   ()"+ .    C   ());
S     .   .       (" .    C   ()"+ .    C   ());
D <I      >  =    D<I      >(10);
S     .   .       ("O      : "+ .  S     ()+"       "+ .   C    ());
```

Résultat:

- a!=b
- a not equals b
- Objet a: A@18d107f classe class A
- a.hashCode()26022015
- b.hashCode()3541984
- c.hashCode()26022015
- Objet x: D@ad3ba4 classe class D

En redéfinissant equals

```
    B
        ;
        ;
B(      ,      )
         . = ;     . = ;

                      (O        )
          (           B)
                ==((B) ).  &&  ==((B) ). ;
                      ;
```

Suite

```
B  =    B(1,2);
B  =    B(1,2);
B  = ;
   ( == )
    S     .   .       (" == ");

    S     .   .       (" != ");
   ( .      ( ))
    S     .   .       ("          ");

    S     .   .       ("              ");
S     .   .       ("o     : "+ .  s     ());
S     .   .       ("o     : "+ .  s     ());
S     .   .       (" .    c   ()"+ .    c   ());
S     .   .       (" .    c   ()"+ .    c   ());
```

Résultat:

- d!=e
- d equals e
- Objet d: B@182f0db
- Objet e: B@192d342
- d.hashCode()25358555
- e.hashCode()26399554

Chapitre IV

Interfaces, classes imbriquées, Object

Chapitre IV

1. Interfaces
2. Classes imbriquées
3. Objets, clonage

classes abstraites

```
            B
                        ();
                         (     )
              =S       .     T    ();
        (      =0; < ; ++)
                    ();
              (S      .     T    () -      );


M   B                       B
                  ()

                                (     )
                    M   B         ().       ( );
```

suite

```
                                (S          )
   S      .    .          ("      ="+
                M   B          .        (1000000));
```

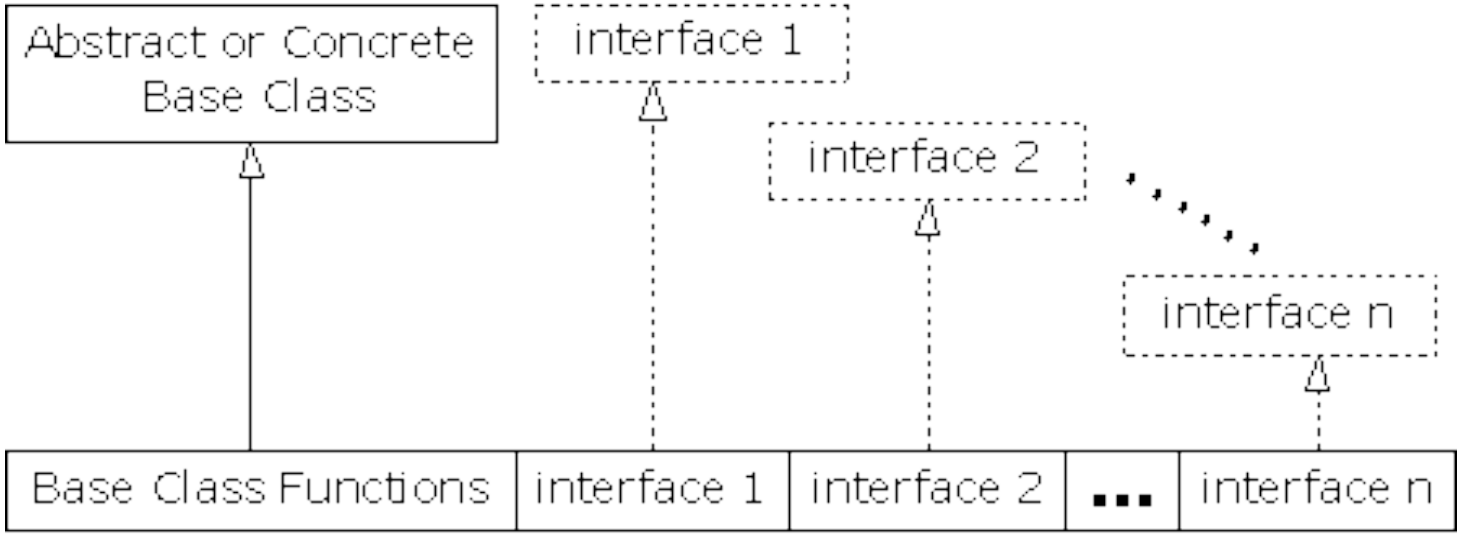
Résultat:

temps=6981893

Interfaces

- Il n'y a pas d'héritage multiple en Java: une classe ne peut être l'extension que d'une seule classe
- Par contre une classe peut implémenter plusieurs interfaces (et être l'extension d'une seule classe)
- Une interface ne contient (essentiellement) que des déclarations de méthodes
- Une interface est un peu comme une classe sans données membres et dont toutes les méthodes seraient abstraites

Héritage "multiple" en java



Exemple:

```
        C          <T>
             T (T    );

  C                  C          <C     >
      , ;
//
                   T (C       )
       ( < . )       1;
             ( . == )
            ( . == )       0;
            -1;
```

Remarques…

- Pourquoi, a priori, l'héritage multiple est plus difficile à implémenter que l'héritage simple?
- Pourquoi, a priori, implémenter plusieurs interfaces ne pose pas (trop) de problèmes?
- (Comment ferait-on dans un langage comme le C?)

Quelques interfaces…

- C : est une interface vide(!) un objet qui l'implémente peut redéfinir la méthode clone
- C : est une interface qui permet de comparer les éléments (méthode T)
- : permet de définir des "threads"
- S : un objet qui l'implémente peut être "sérialisé" = converti en une suite d'octets pour être sauvegarder.

Déclarations

- une interface peut déclarer:
  - des constantes (toutes les variables déclarées sont et)
  - des méthodes (elles sont implicitement)
  - des classes internes et des interfaces

Extension

les interfaces peuvent être étendues avec extends:

- Exemple:

  public interface SerializableRunnable
  extends Serializable, Runnable;

  (ainsi une interface peut étendre de plusieurs façons une même interface, mais comme il n'y a pas d'implémentation de méthodes et uniquement des constantes ce n'est pas un problème)

Exemple

```
        X
         =0;

                 X
         =1;
           =   +X.   ;



           I        H
                             (S          )
      S      .    .         (" .   ="+ .   +"  .     ="+ .     );
         =      ();
          S      .   .        (" .   ="+ .   +
                   " (( ) ).   ="+(( ) ).   +
                   " ((X) ).   ="+((X) ).   );
```

Z.val=1 Z.somme=1

z.val=1 ((Y)z).val=1 ((X)z).val=0

Redéfinition, surcharge

A
();
();

B
();
( );
();

C A,B

Rien n'indique que les deux méthodes () ont la même "sémantique". Comment remplir le double contrat?

Chapitre IV

1. Interfaces
2. Classes internes et imbriquées
3. Object, clonage

Classes imbriquées (nested classes)

- Classes membres statiques
  - membres statiques d'une autre classe
- Classes membres ou classes internes (inner classes)
  - membres d'une classe englobante
- Classes locales
  - classes définies dans un bloc de code
- Classes anonymes
  - classes locales sans nom

Classe imbriquée statique

- membre statique d'une autre classe
  - classe ou interface
  - mot clé
  - similaire aux champs ou méthodes statiques: n'est pas associée à une instance et accès uniquement aux champs statiques

Exemple

```
P   C
                          C
                C             S      ();
                          S      (C              );

  C             ;
                      (C          )
        .   S       (    );
          = ;

          O             ()
       C              ;
          (!   V    ())
             =     ;
              =     .   S       ();
                  ;

                      ;

                    V    ()
                ==     ;
```

exemple (suite)

```
    E     C                          P    C       .C
        ;
          E     C          (     )      . = ;
  P   C       .C                  ;
          P   C       .C                S       ()
                    ;

                    S       (P   C       .C           )
          = ;

                  ()          ;
```

et le main

```
                    (S              )
P   C          ;
E     C             ;
 =    P   C         ();
   (      =0;  <12; ++)
     =    E     C         ( );
     .         ( );

     (! .   V   ())
    S     .   .       (
             ((E     C         ) .        ()).   ());
```

Remarques

- Noter l'usage du nom hiérarchique avec '.'
- On peut utiliser un import:
  - P C .C ;
  - P C ;

(Exercice: réécrire le programme précédent sans utiliser de classes membres statiques)

Classe membre

- membre non statique d'une classe englobante
- peut accéder aux champs et méthodes de l'instance
- une classe interne ne peut pas avoir de membres statiques
- un objet d'une classe interne est une partie d'un objet de la classe englobante

Exemple

```
C       B
                    ;
                      ;
        A           ;
              A
            S           ;
                          ;
    A       (S            ,              )
              .     =     ;
              .           =          ;

            S           S       ()
                          "+":"+    +" "+          ;
```

Suite

```
//
                    (              )
            +=              ;
        =   A       ("       ",         );

                     (              )
            -=              ;
        =   A       ("         ",         );
```

Remarques

- dans toString
- this:
  - der=this.new Action(…);
  - CompteBancaire.this.numero

Classe interne et héritage

```
   E
       I

   E       E                  E
       I       E                  I
I          =     I       E        ();

   A                E       .I
A     (E           )
      .      ();
```

(un objet Interne (ou d'une de ses extensions) n'a de sens qu'à l'intérieur d'un objet Externe)

Quelques petits problèmes

```
X
    ;
     H
                    ()   ++;
```

Si i est une donnée membre de Y… c'est ce i qui est incrémenté

X.this.i et this.i lèvent cette ambiguïté.

Suite

```
H
        ()
        (    )
   I
            ()  ;
           ()
           ();
     H.    .     ();
  // print(1); tous les print sont occultés
```