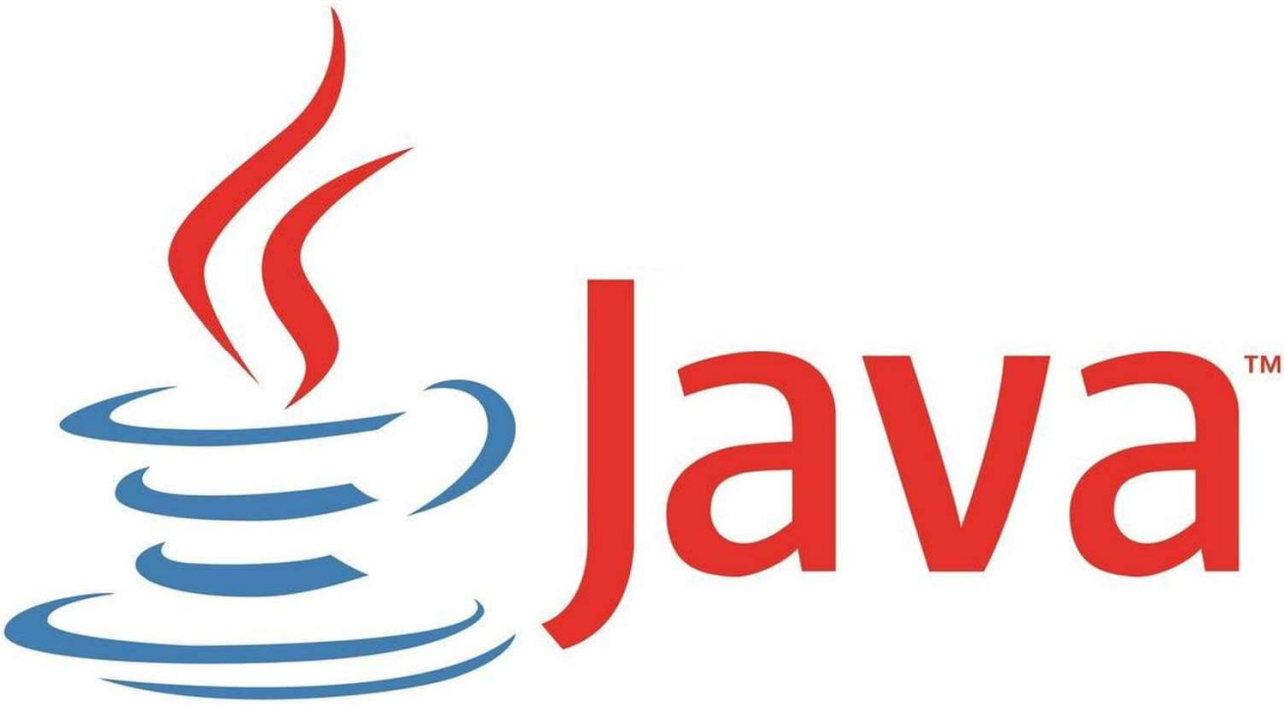
**Lescop Nolann**
**1D1**

Vannes
iut :
Université Bretagne Sud

# TP N°3 – R2.01



**Lescop Nolann**
**1D1**

## Table des matières

**Lescop Nolann**
**1D1**

## Question 1 :

c) On remarque que les classes sont triés dans des packages avec javac, il y a le package client pour la classe C et E ainsi que le package jeux pour les classes A, B, D.

d) Lorsque on exécute la class E, on obtiens l'affichage suivant :

```
nolann@nolann-ASUS-TUF-Gaming-F15-FX506HM-FX506HM:~/Documents/Info/BUT/BUT_1/Initiation_au_développement/R2.01/R2.01_P1/Tps/TP3/ws$ java client.E
Constructeur de A
Constructeur de A
Constructeur de B
Constructeur de A
Constructeur de B
Constructeur de C
Constructeur de A
```

## Question 2 :

a) Les chaînages :
- de B et A : Il n'y a pas de paramètre dans le constructeur de la class A, le chaînage est donc fait automatiquement à la compilation en première instruction du constructeur de A.
- de C et B : Le chaînage est fait manuellement avec le mot clé super() en première instruction du constructeur car le constructeur de la class B a un paramètre dans son constructeur.
- de D et A : Il y a un constructeur qui est crée à la compilation et le chaînage est fait automatiquement à la compilation car A n'a pas de paramètre.
- de E avec Object. : le chaînage est fait automatiquement par le compilateur.
- De A avec Object : chaînage automatique vers la super class E (réalisé par le constructeur).

b) Si le mot clé super() est mis en commentaire, il y a une erreur à l'exécution :

```
nolann@nolann-ASUS-TUF-Gaming-F15-FX506HM-FX506HM:~/Documents/Info/BUT/BUT_1/Initiat
.01/R2.01_P1/Tps/TP3/ws$ javac -d ../class ../src/*.java
../src/C.java:5: error: constructor B in class B cannot be applied to given types;
    public C(){
               ^
  required: int
  found:    no arguments
  reason: actual and formal argument lists differ in length
1 error
```

c) le compilateur nous dit que le mot clé super() doit être placé en première instruction :

```
nolann@nolann-ASUS-TUF-Gaming-F15-FX506HM-FX506HM:~/Documents/Info/BUT/BUT_1/Initia
.01/R2.01_P1/Tps/TP3/ws$ javac -d ../class ../src/*.java
../src/C.java:7: error: call to super must be first statement in constructor
        super(-10);
             ^
1 error
```

## Question 3 :

this.a1=1; : la class C n'a pas hérité avec usufruit de cette instruction
this.a2=2; la class C n'a pas hérité avec usufruit de cette instruction
this.a3=3; : la class C a hérité avec usufruit de cette instruction
this.a4=4; : la class C a hérité avec usufruit de cette instruction

Résultat compilation :

```
nolann@nolann-ASUS-TUF-Gaming-F15-FX506HM-FX506HM:~/Documents/Info/BUT/BUT_1/Initiation_
.01/R2.01_P1/Tps/TP3/ws$ javac -d ../class ../src/*.java
../src/C.java:13: error: a1 has private access in A
        this.a1=2;
            ^
../src/C.java:14: error: a2 is not public in A; cannot be accessed from outside package
        this.a2=3;
            ^
2 errors
```

Obtention de la valeur A1 et affichage :

```
getA1 de A
1
m2 de B
m2 de A
```

b)

```
this.a1=2; la class B n'a pas hérité avec usufruit de cette instruction
this.a2=3; la class B a hérité avec usufruit de cette instruction
this.a3=4; la class B a hérité avec usufruit de cette instruction
this.a4=5; la class B a hérité avec usufruit de cette instruction
```

Erreur à la compilation de la class B :

```
../src/jeux/B.java:14: error: a1 has private access in A
        this.a1=2;
            ^
1 error
```

## Question 4 :

a)

Résultat à la compilation :

```
../src/client/E.java:14: error: a1 has private access in A
        unC.a1 = 3;
           ^
../src/client/E.java:20: error: a1 has private access in A
        unC.a1 = 3;
           ^
../src/client/E.java:21: error: a2 is not public in A; cannot be accessed from outside package
        unC.a2 = 4;
           ^
../src/client/E.java:22: error: a3 has protected access in A
        unC.a3 = 5;
           ^
4 errors
```

```
// Accès inter-objets direct
unC.a1 = 3; : Accès impossible car a1 est privée dans A
unC.m7();
unC.m8();
unC.m1();

// Accès inter-objets indirect
unC.a1 = 3; : Accès impossible pour la class C car a1 est privée dans A
unC.a2 = 4; : Accès impossible pour la class C car a2 n'est pas public dans A
unC.a3 = 5; : Accès impossible pour la class C car a1 est privée dans A
unC.a4 = 6;
```

b)

Résultat à la compilation :

```
../src/client/E.java:27: error: a1 has private access in A
        unA.a1 = 3;
           ^
../src/client/E.java:28: error: a2 is not public in A; cannot be accessed from outside package
        unA.a2 = 4;
           ^
../src/client/E.java:29: error: a3 has protected access in A
        unA.a3 = 5;
           ^
3 errors
```

```
// Accès inter-objets entre 2 classes de package différents
unA.a1 = 3; : Accès impossible car a1 est privée dans A.
unA.a2 = 4; : Accès impossible car a2 n'est pas dans le même package et a2 n'est pas public.
unA.a3 = 5; : Accès impossible car a3 est protégée dans A.
unA.a4 = 6;
```

c)
Tableau 1 : Le tableau 1 est correct et correspond aux erreurs
Tableau 2 : Le tableau 2 est correct et correspond aux erreurs
Tableau 3 : Le tableau 3 est correct et correspond aux erreurs

## Question 5 :

Résultat de la compilation :

```
nolann@nolann-ASUS-TUF-Gaming-F15-FX506HM-FX506HM:~/Documents/Info
../src/client/E.java:33: error: m2() has protected access in A
        unA.m2();
           ^
../src/client/E.java:34: error: m2() has protected access in B
        unB.m2();
           ^
../src/client/E.java:35: error: m2() has protected access in B
        unC.m2();
           ^
3 errors
```

Le code :

```
// unA.m2();
// unB.m2();
// unC.m2();
unD.m2();
```

unA.m2() ; :
Cela ne marche pas car m2 est définis comme protected dans la class A donc cela signifie que la class E ne peux pas y accéder car elle dans un autre package.

unB.m2() ; :
Cela ne marche pas pour la même raison que la class A.

unC.m2() ; :
la class C n'a pas de m2 donc elle hérite du m² de la class B sauf que la le m2 de B est protected donc ça ne marche pas.

unD.m2 ; :
Cela fonctionne car la methode m2 est redéfinis dans la class B en public.

## Question 6 :

a)
unA=unB;
B est une sous-classe de A donc B < A donc il n'y a pas d'erreur.

unA=unD;
D est une sous-classe de A donc D < A donc il n'y a pas d'erreur.

unB=unC;
C est une sous-classe de B donc C < B donc il n'y a pas d'erreur.

unA=unC;
C est une sous-classe de B et B est une sous-classe de A donc C < A donc il n'y a pas d'erreur.

unA.m5();
m5 n'existe pas dans A mais m5 existe dans C sauf que comme A > C cela ne fonctionne pas.

unC=unA;
Il y a une erreur car C n'est pas supérieur à A cependant l'inverse fonctionne.

b)
Le compilateur Java ne travaille que sur le type statique des variables.
- unA a un type déclaré A.
- m5() n'existe pas dans A, donc le compilateur interdit l'appel à cette méthode.
Imaginons que unA contienne en réalité un objet C qui a bien une méthode m5().
- À l'exécution, l'appel unA.m5(); aurait fonctionné.
- Mais Java ne peut pas garantir que unA contient bien un C (il pourrait aussi contenir un B ou un D, qui n'ont pas m5()).
Le compilateur interdit donc unA.m5(); par précaution.

c)
unA=unC; : Compile

unC=(C)unA; : Compile

unC.m5(); : Compile

unA=unD; : Compile

unC=(C)unA; : Compile

Résultat exécution :

```
getA1 de A
1
m2 de B
m2 de A
Exception in thread "main" java.lang.ClassCastException: class jeux.D cannot be cast to class client.C (jeux.D and client.C are in unnamed module of loader 'app')
        at client.E.main(E.java:44)
```

Lors de l'exécution, on constate que la dernière instruction ne fonctionne pas car on essaye de forcer (cast) C mais C n'est pas un D donc java détecte une erreur à l'exécution.

Conclusion :
Le transtypage peut permettre d'accéder à des méthodes spécifiques, mais il doit être utilisé avec précaution afin d'éviter de forcer un objet en un certain type alors qu'il n'est pas réellement de ce type.
Le transtypage n'est pas vérifié à la compilation mais à l'exécution.