IPO- Null, Wrapper

Alice Jacquot

null

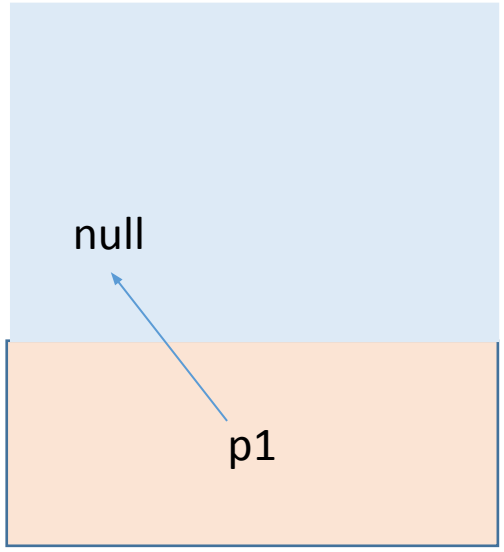
La valeur ***null*** est la valeur par défaut des variables de type objet (= non primitif) non instanciés :

String s; est équivalent à String s = null;

Elle indique qu'il n'y a pas eu d'objet alloué correspondant dans le tas (heap).

Comment se crée une instance ?

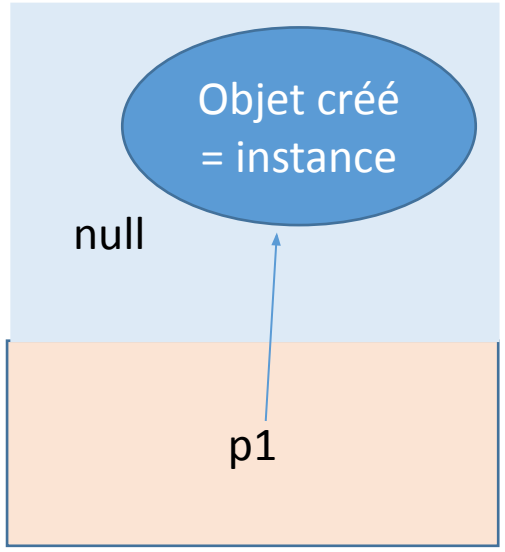
```
String str1; // declaration str1 est ici null
str1="toto"; //affectation
```



étape1

Heap managé

stack



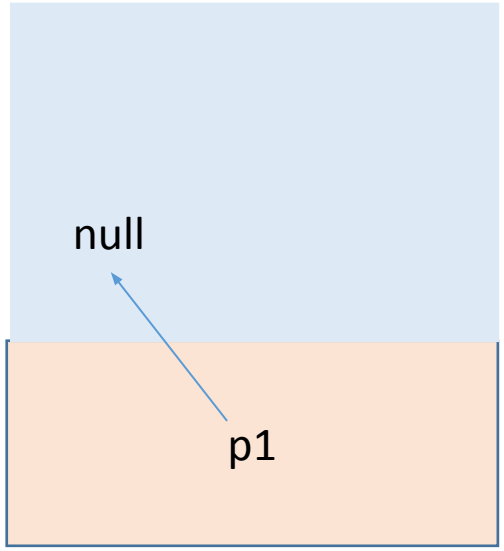
Étape 2

Heap managé — Le garbage collector y supprime les instances vers lequelles ne pointent aucune référence

stack — Les objets s'y dépile « tout seul » à la fin du programme/méthode concerné

Comment se crée une instance ?

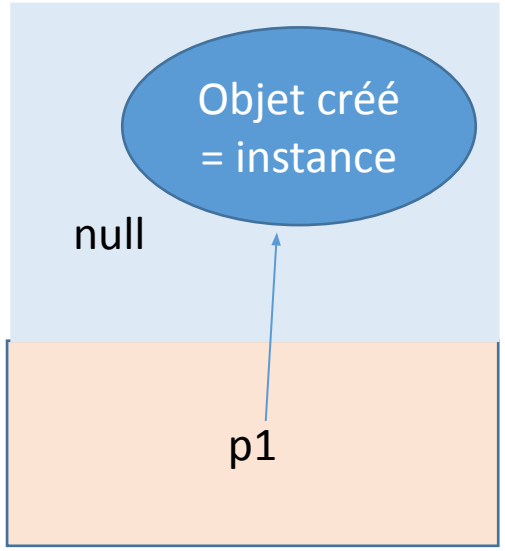
```
String str1; // declaration str1 est ici null
str1="toto"; //affectation

Personne p1; //declaration : p1 de type Personne, une classe que l'on aura définie
// etape 1. Idem à Personne p1 = null
p1=new Personne(); //instanciation etape 2
p1=null; // retour au même état qu'apres etape 1, G.C. supprime l'objet (pas tjrs immédiat)
```



Heap managé

stack

étape1



Étape 2

Heap managé — Le garbage collector y supprime les instances vers lequelles ne pointent aucune référence

stack — Les objets s'y dépile « tout seul » à la fin du programme/méthode concerné

Un usage : tester la création

On teste souvent si un objet est à null pour savoir s'il a été créé ou s'il existe.

Par exemple, avec map de type HashMap<String, ArrayList<String>> :

```
ArrayList<String> textsInSameLanguage = map.get(lang);
            if (textsInSameLanguage == null){
                textsInSameLanguage = new ArrayList<>();
                map.put(lang,textsInSameLanguage);
            }
            textsInSameLanguage.add(text);
```

Un usage : tester la création

On teste souvent si un objet est à null pour savoir s'il a été créé ou s'il existe.

Par exemple, avec map de type HashMap<String, ArrayList<String>> :

```
ArrayList<String> textsInSameLanguage = map.get(lang);
                  if (textsInSameLanguage == null){
                      textsInSameLanguage = new ArrayList<>();
                      map.put(lang,textsInSameLanguage);
                  }
                  textsInSameLanguage.add(text);
```

S'il n'y a pas déjà un tableau correspondant à cette clé, on le crée

NullPointerException

null n’est d’aucun type : il ne connaît aucune méthode !

```
Personne p1=null;
System.out.println(p1.getNom()+" "+p1.getPrenom());
```

Déclenche un erreur à l’**exécution** (compile bien) :

```
Exception in thread "main" java.lang.NullPointerException
at debut.TestPersonne.testinstanciation(TestPersonne.java:22)
at debut.TestPersonne.main(TestPersonne.java:6)
```

Signifie que l’on tente d’utiliser un objet non-instancié :
On ne peut pas y accéder !

Valeurs par défaut

Pour toutes les **classes**, la valeur par défaut est **null** : à la définition de la variable, sans instanciation, elle sera affectée à null.

Les noms de types commencent alors par des **majuscules** (**S**tring, **H**ashMap, **L**ocale,...)

Pour les **types primitifs** (int, long, short, byte, float, double, char, boolean) la valeur par défaut est **0 (false pour boolean)**.

Ce sont **les seuls noms de types** commençant par une **minuscule**.

Valeurs par défaut

Pour toutes les **classes**, la valeur par défaut est **null** : à la définition de la variable, sans instanciation, elle sera affectée à null.

Les noms de types commencent alors par des **majuscules** (**S**tring, **H**ashMap, **L**ocale,...)

Pour les **types primitifs** (int, long, short, byte, float, double, char, boolean) la valeur par défaut est **0 (false pour boolean)**.

Ce sont **les seuls noms de types** commençant par une **minuscule**.
Cependant…

variable b might not have been initialized

La valeur par défaut ne sert que les attributs… (cf suite du cours)

Les Wrappers

Un Wrapper est une classe encapsulant un type primitif : Integer, Double, Boolean…

Historiquement, les instances se créent à la manière des objets (ce **sont** des objets !) :
Integer a = new Integer (5);

On préfère aujourd'hui l'écriture :

Integer a = 5;

En soi, c'est un "abus de langage"

Wrapper : des méthodes !

Les wrappers sont des classes, et présentent une interface avec des méthodes utilisables (cf javadoc)

```
Integer a = 5;
long l = a.longValue;
```

Ils possèdent aussi de nombreuse méthodes statiques, qui servent aux usage courant sur ces types

```
int Integer.parseInt(String s)
```

Valeurs par défaut

Les wrapper sont des objets : leur valeur par défaut est **null**

```
Integer a; // a vaut null
```

Ce peut être utile pour différencier une variable non initialisée dans certains cas d'une variable à 0 (ou -1, ou autre valeur spéciale).