



## PREAMBULE

P.Morat

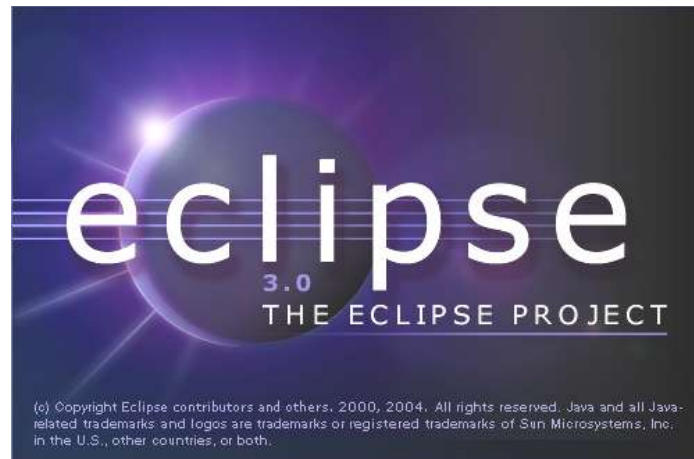
Un environnement de développement tel que « Eclipse » fournit une plateforme de construction de produits logiciels ou autres que l'on désigne sous l'acronyme IDE. Ce type de plateforme est par définition extensible pour permettre d'intégrer aisément de nouveaux outils spécifiques aux produits que l'on souhaite élaborer avec celle-ci. L'utilisation que l'on va faire de cet environnement est restreint au développement d'applications informatiques en Java. Dans ce cas « Eclipse » fournit une « vision » spécifique de l'ensemble de l'information manipulable. Cette vision tient compte des spécificités de Java et de la structuration des informations élaborées lors d'un développement d'une application avec ce langage. Il en serait différemment si l'on utilisait un autre langage, par exemple C ou C++ (voire ADA). Le cadre du développement d'application informatique n'est donc qu'une utilisation particulière d'«Eclipse» dans un ensemble bien plus vaste de possibilités.

De ce fait, l'environnement doit permettre de choisir la vision que l'on souhaite utiliser. Cette vision (**Perspective**) fixe la façon dont est visionnée l'information et les opérateurs (actions) que l'on peut effectuer sur ces informations. Le concept de Perspective dans «Eclipse» offre cette diversité. Chaque perspective est identifiée par un nom pour permettre sa sélection. Elle est constituée de vues (**View**) qui sont les « éditeurs » utilisables dans ce cadre et de menus ou autres moyens donnés à l'utilisateur pour effectuer des opérations. Chaque perspective est personnalisable permettant à chacun de constituer une organisation de son ***bureau d'étude***.

Les informations manipulées dans le cadre d'une activité de développement peuvent être très nombreuses et une vision unique ne saurait être suffisante car trop complexe pour être utilisée raisonnablement. Dans ce cas plusieurs perspectives peuvent être élaborées pour permettre d'appréhender cette information selon les différentes facettes souhaitées. Dans l'utilisation que nous allons faire d'«Eclipse» nous utiliserons essentiellement les perspectives attachées au développement Java (Browser de classes Java, Debugger de programmes Java, Navigateur de système de fichiers, ...).

«Eclipse» peut être considéré comme un ***méta-éditeur*** car il propose la possibilité de modéliser l'éditeur spécifique que l'on souhaite mettre en œuvre pour une activité particulière de développement. Chaque utilisation d'«Eclipse» correspond à une activité bien identifiée utilisant de ce fait la(es) perspective(s) appropriée(s). Une activité pouvant être exécutée de façon intermittente sur une longue période, il s'ensuit qu'il est indispensable de conserver la configuration (**Projet**) avec laquelle elle est développée. Le concept de projet dans «Eclipse» fixe l'ensemble de l'information attaché à celui-ci ainsi que la configuration de développement que l'utilisateur a constitué pour ce développement particulier. Ainsi deux développements dans un même type d'activité peuvent avoir des configurations légèrement différentes. Cette personnalisation permet à chacun d'adapter son bureau d'étude à sa façon de travailler ou de différencier son environnement de travail en fonction de la spécificité de l'activité (Application Standalone, Application Web, ...).

«Eclipse» n'est pas le seul environnement offrant ces possibilités, notre choix c'est porté sur lui car il est utilisable gratuitement dans notre environnement universitaire, il était un des premiers à fournir certaines fonctionnalités qui nous apparaissaient indispensables de contoyer comme les fonctions de restructuration du logiciel (**refactoring**). L'environnement **NetBeans** et d'autres offrent sensiblement les mêmes fonctionnalités, ils diffèrent relativement peu dans le principe, un peu sur l'interface utilisateur proposée et parfois sur les outils préexistants fournis.



# Créer une simple application Java avec ECLIPSE

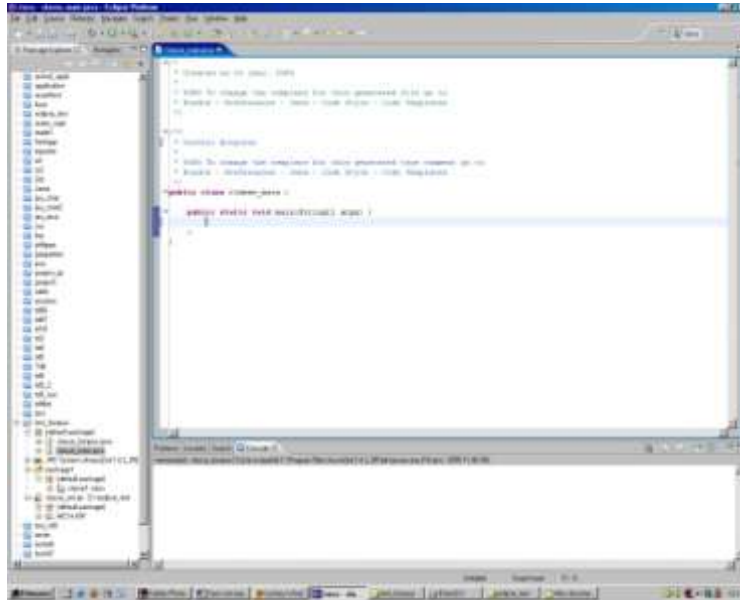
Par Ahcène BOUNCEUR

## INTRODUCTION

Dans cet exemple nous allons vous montrer comment peut-on créer une simple application Java en utilisant l'IDE Eclipse. Cette application doit tout simplement afficher la chaîne de caractères "Bonjour".

### Etape 1 : Création d'un nouveau projet

Lancez Eclipse.



Vous devez créer un projet Java dans lequel se trouveront vos classes. Choisissez le bouton Nouveau (ou New) et cliquez sur la flèche-bas à sa droite puis sur Projet (ou Project), voir Figure 1-a, ou dans le menu Fichier → Nouveau → Projet (ou File → New → Project), voir Figure 1-b.

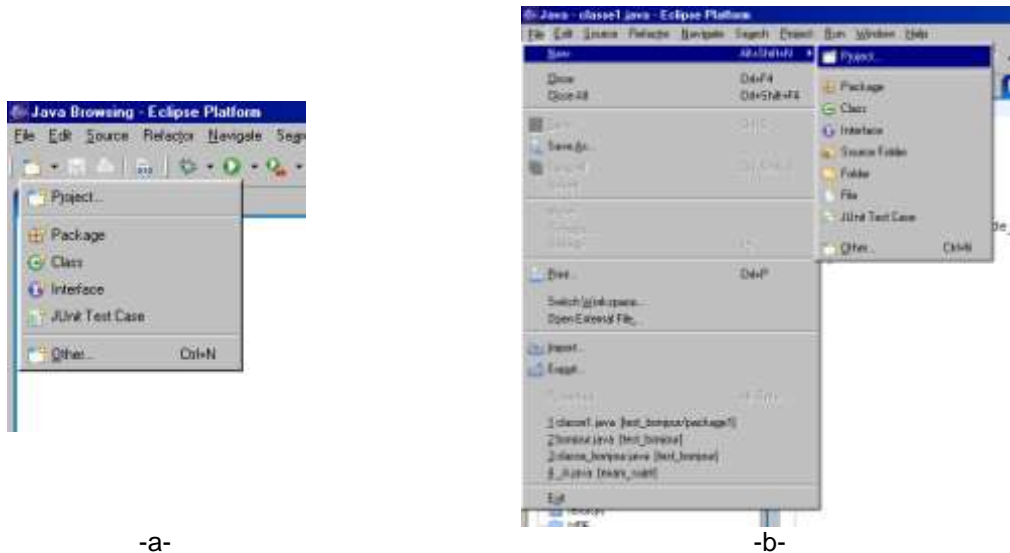


Figure 1

Puis une fenêtre s'affiche (voir Figure 2), dans cette fenêtre spécifiez le type du projet à créer. Vous choisissez donc, Projet Java (ou Java Project).

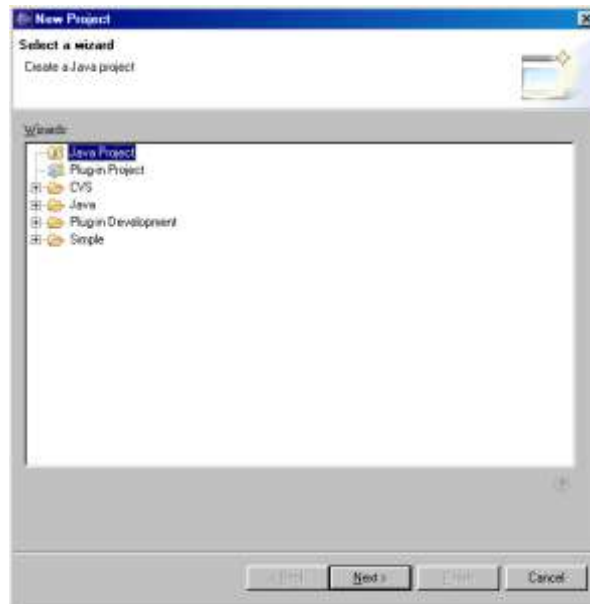


Figure 2

Puis cliquez sur le bouton **Suivant** (ou **Next**), une autre fenêtre s'affiche (voir Figure 3), dans laquelle il faut entrer le nom du projet dans la partie **nom du projet** (ou **Project Name**), et faites entrer le nom "test\_bonjour":

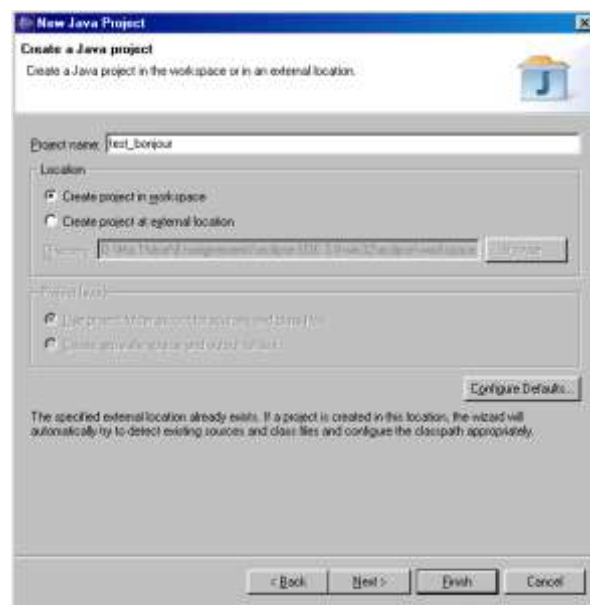
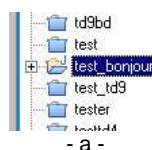
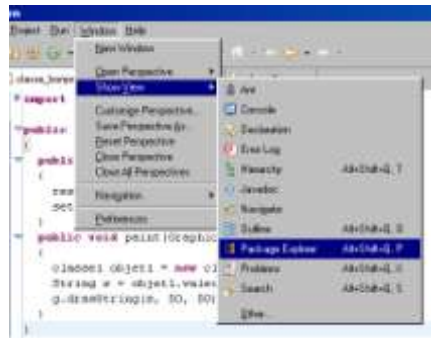


Figure 3

Puis cliquez sur le bouton **Fin** (ou **Finish**). Le projet est créé (voir Figure 4-a). Si vous ne voyez pas les projets comme dans la Figure 4-a alors cliquez dans le menu sur **Fenêtre** (ou **Window**) puis sur **Afficher** (ou **Show View**) puis sur **Explorateur des packages** (ou **Package Explorer**), voir Figure 4-b.





- b -

Figure 4

**Question 1 :** Que signifie un projet créé, au niveau du disque-dur ?

**Exercice 1 :** Comment peut on connaître le chemin où se trouve le projet ?

## Étape 2 : Création de la classe principale

La classe principale en Java est la classe qui contient la méthode `main` que vous pouvez écrire vous-même, comme elle peut être créée automatiquement en créant une nouvelle classe. Pour créer une nouvelle classe, cliquez sur le projet que vous avez créé (`test_bonjour`) avec le bouton droit de la souris (ou bien dans le menu sur **Projet** ou bien **Project**) puis un sous menu s'affichera, choisissez **Nouveau** (ou **New**), puis un autre sous menu s'affichera, choisissez donc **class** (voir Figure 5).

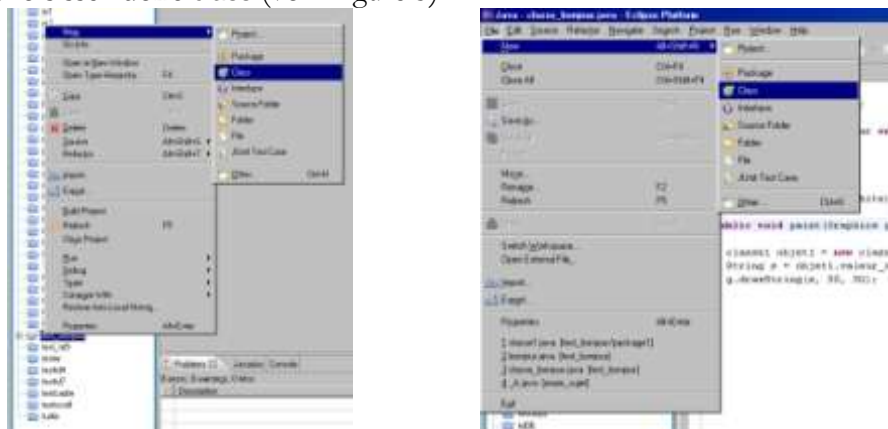


Figure 5

Une fenêtre s'affichera dans laquelle il faut entrer le nom de la classe. Faites entrer le mot "`classe_bonjour`" dans la partie **Nom** (ou **Name**) tel que le montre la Figure 6.



Figure 6

Puis cocher la case **public static void main(String[] args)** pour créer automatiquement la méthode **main** et puis cliquez sur le bouton **Fin** (ou **Finish**). La classe sera donc créée, c'est un fichier appelé **classe\_bonjour.java** qui est créé dans votre projet (voir Figure 7).

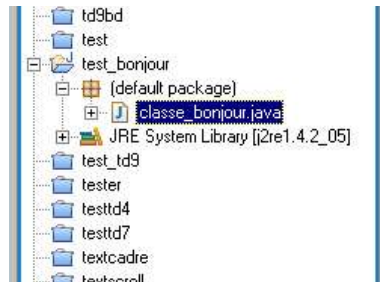


Figure 7

Un **package** est un répertoire créé dans votre projet, et la Figure 7 montre bien que la classe **classe\_bonjour** se trouve dans un package nommé (**default package**) ; vérifiez dans votre disque si un tel répertoire a été créé.

**Question 2 :** Un tel répertoire existe-il ou non ? Où est ce que se trouve donc le fichier source **classe\_bonjour.java** dans votre disque ? Conclusion ?

### Étape 3 : Ecriture du contenu de la classe **classe\_bonjour**

Le contenu du fichier **classe\_bonjour.jar** est affiché dans une fenêtre texte (voir Figure 8).

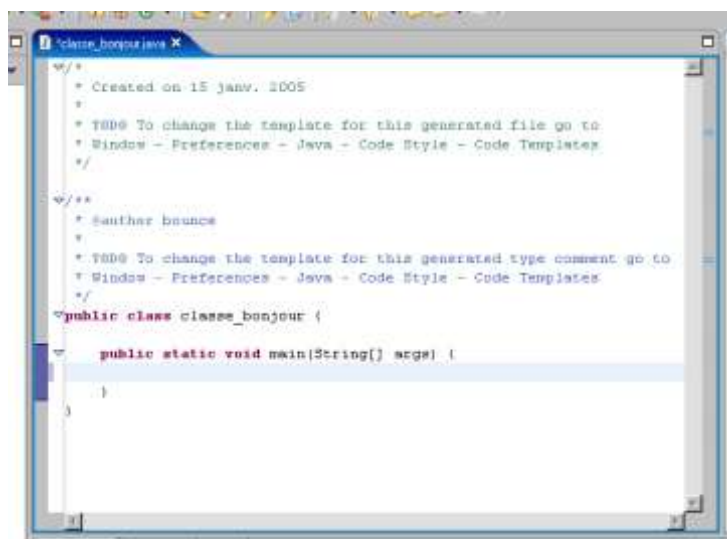


Figure 8

Nous allons donc modifier le contenu du code source **classe\_bonjour.java** en ajoutant un code permettant d'afficher "Bonjour" à l'intérieur de la méthode **main** (voir Figure 9).

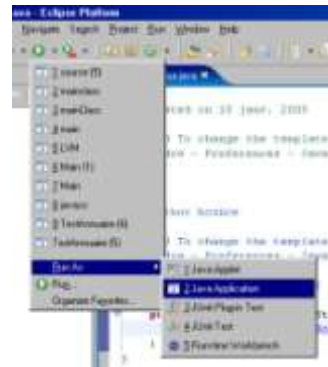
```
public static void main(String[] args) {
    System.out.println("Bonjour");
}
```

Figure 9

Pour compiler ce code, cliquez sur le projet avec le bouton droit de la souris puis sur **compiler le projet** (ou **Build Project**). Et pour l'exécuter, glissez le pointeur de la souris sur le bouton **Exécuter** (ou **Run**), voir Figure 10-a, et cliquez sur la flèche-bas à sa droite puis sur **Exécuter comme** (ou **Run As**) puis sur **Java Application** (voir Figure 10-b).



-a-



-b-

Figure 10

**Question 3 :** Observez le résultat de l'exécution puis donnez le nom de la fenêtre qui affiche le résultat et ainsi que le résultat.

## Étape 4 : La classe classe\_bonjour sous forme d'une applet

Supprimez tout le contenu de la classe class\_bonjour.java, puis réécrivez le code présenté dans la Figure 11.

```
import java.awt.*;

public class classe_bonjour extends java.applet.Applet
{
    public void init()
    {
        resize(300,80);
        setBackground(Color.white);
    }
    public void paint(Graphics g)
    {
        g.drawString("Bonjour", 50, 50);
    }
}
```

Figure 11

Exécuter maintenant.

**Question 4 :** Que ce qu'il faut faire pour exécuter cette applet ?

**Question 5 :** Le résultat est il le même que celui d'une application ? Où est ce que le mot "Bonjour" est affiché ?

**Exercice 2 :** Créez une page-web qui utilise cette applet.

## Étape 5 : Création et utilisation d'un package

Pour créer un package, cliquez sur le projet avec le bouton droit puis sur nouveau puis sur package ou en utilisant le menu (voir Figure 12).

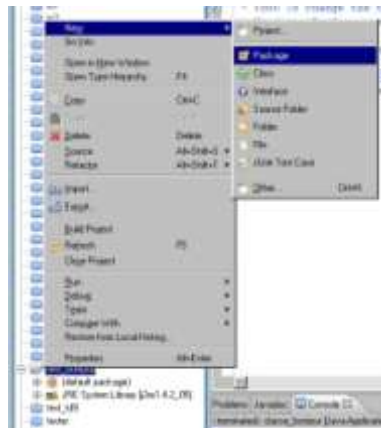


Figure 12

Une fenêtre s'affiche.



**Question 6 :** Faite entrer, dans cette fenêtre, comme nom le mot “package”, est il possible ? Sinon, entrer le nom “package1”

**Question 7 :** Pourquoi le mot “package” ne peut pas être un nom de notre package ?

Le résultat est présenté dans la Figure 13.



Figure 13

Créez une classe appelée “classe1” dans le package “package1”.

**Question 8 :** Quelles sont les démarches à suivre pour créer cette classe ?

Créez dans la classe `classe1` un attribut appelé “s” de type `String` et un constructeur qui initialise la variable `s` à la valeur “Salut” et une méthode appelée `valeur_de_s` qui renvoie la valeur de `s`.

**Question 9 :** Que faut-il écrire ?

Modifiez l'applet en affichant à la place de “bonjour” la valeur de l'attribut `s` d'un objet de type `classe1`.

**Question 10 :** Que faut-il écrire ? Exécutez, et dire ce qui est affiché.

Vous avez sûrement ajouté la commande `import package1.classe1`, sinon erreur.

## Étape 6 : CLASSPATH

### 1 – Appel d'une classe dans un package local

Cliquez dans le menu sur **Projet** (ou **Project**) puis sur **Propriétés** (ou **Properties**) une fenêtre s'affiche, regardez la partie gauche de cette fenêtre et cliquez sur **Java Build Path** puis dans la partie droite de cette fenêtre cliquez sur l'onglet **Librairies** (ou **Libraries**), vous devez obtenir le résultat de la Figure 14.

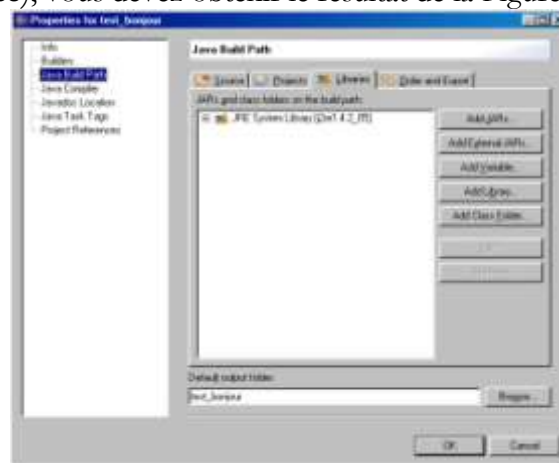


Figure 14

Cliquez ensuite sur le bouton **Add Class Folder**, une autre fenêtre s'affiche (voir Figure 15).



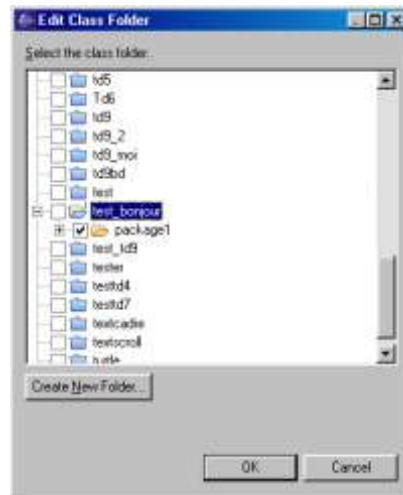


Figure 15

Cochez la case à gauche de **package1**, puis cliquez sur OK. Une fenêtre s'affiche pour confirmer l'ajout du package (voir Figure 16).



Figure 16

Cliquez donc sur OK. Et le résultat : le package est ajouté (voir Figure 17).

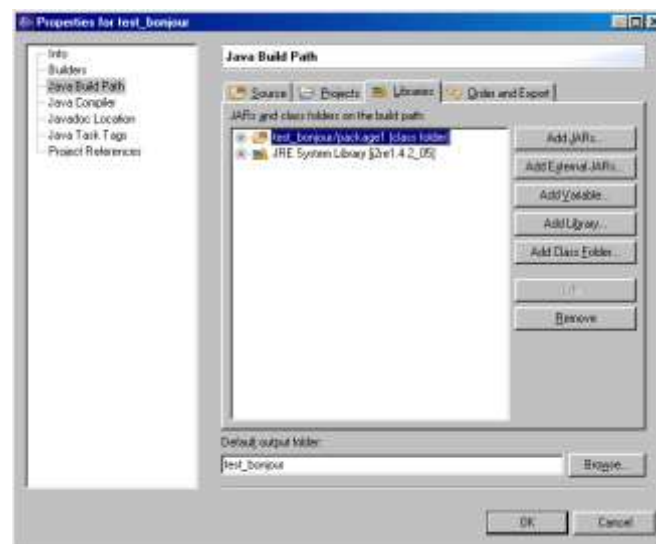


Figure 17

Alors :

**Question 11 :** La commande `import package1.classe1` est elle toujours nécessaire ?

**Question 12 :** Enlevez cette commande (ou ligne) puis exécutez, c'est quoi le résultat ?

## 2 – Appel d'une classe dans un JAR

Il existe deux procédures différentes pour appeler un JAR. La première concerne un JAR qui se trouve dans le projet même et la deuxième, concerne un JAR qui se trouve à l'extérieur du projet (c'est-à-dire dans n'importe quel endroit de votre disque).

La première procédure est exactement similaire à celle qui ajoute un package, précédemment décrite (étape 6, section 1), seulement au lieu de cliquer sur le bouton **Add Class Folder**, cliquez sur le bouton **Add JARs** puis toute la liste des JARs locaux s'affichera. Pour ajouter un JAR se trouvant dans un autre endroit que celui de votre projet, procédez comme suit :

1. D'abord créez un répertoire dans votre disque et appelez le **mes\_jars**.
2. Télécharger le fichier **classe\_ext.jar** sur l'adresse suivante : <http://>
3. Placez ce fichier dans le répertoire **mes\_jars** que vous venez de créer.
4. Dans Eclipse, répétez la première démarche de l'étape 6 jusqu'à ce que la fenêtre **Propriétés** ou **Properties** (voir Figure 14) s'affiche. Cliquez donc sur le bouton **Add External JAR...**
5. spécifiez le fichier JAR que vous venez de télécharger.

**Question 13 :** Quelles sont les classes que contient ce JAR ?

**Question 14 :** Ce JAR contient une classe appelée **classe\_plus**, quels sont les attributs de cette classe ? Ils sont de quels types ?

**Question 15 :** Quelles sont les différents constructeurs et les différentes méthodes de cette classe ? Les méthodes sont de quels types ?

**Question 16 :** La classe **classe\_plus** contient uniquement un seul attribut de type **String**. Utilisez votre applet pour afficher la valeur de cet attribut. Quelle est donc la valeur de cet attribut ?

### 3 – Créer un JAR

Créez dans votre projet (dans la racine) une classe, donnez lui le nom **classe2**. Supposons maintenant que nous voulons créer un JAR qui contiendra à priori la classe **classe2**. Cliquez sur **classe2** dans l'explorateur des packages avec le bouton droit de la souris puis choisissez **Exporter** (ou **Export**).

**Exercice 3 :** Décrivez toutes les étapes à suivre pour créer ce JAR dans le répertoire **mes\_jars**.

**Exercice 4 :** Créez un JAR qui contiendra tout le projet que vous venez de créer. Puis créez un nouveau projet qu'il faut appeler **projet\_jar**. Décrivez les démarches à suivre pour exécuter l'applet qui se trouve dans ce JAR.

## Étape 8 : Le Javadoc

Il existe deux types de commentaires dans les codes Java. Le premier type représente les commentaires classiques qui s'écrit comme suit :

```
/*
Ligne 1 de mon commentaire,
Ligne 2 de mon commentaire,
...
*/
```

Ce type de commentaire est utile pour le programmeur. Le deuxième type a le même principe que le premier mais il est utile pour la génération du Javadoc. Il s'écrit comme suit :

```
/**
Ligne 1 de mon commentaire,
Ligne 2 de mon commentaire,
...
*/
```

Chaque commentaire de type 2 doit définir la méthode le suivant.

Ajouter à votre programme tous les commentaires de type 2 définissant toutes les méthodes et attributs.

**Question 17 :** Comment générer un Javadoc pour cette application ?



# Créer une simple application Java avec NetBeans

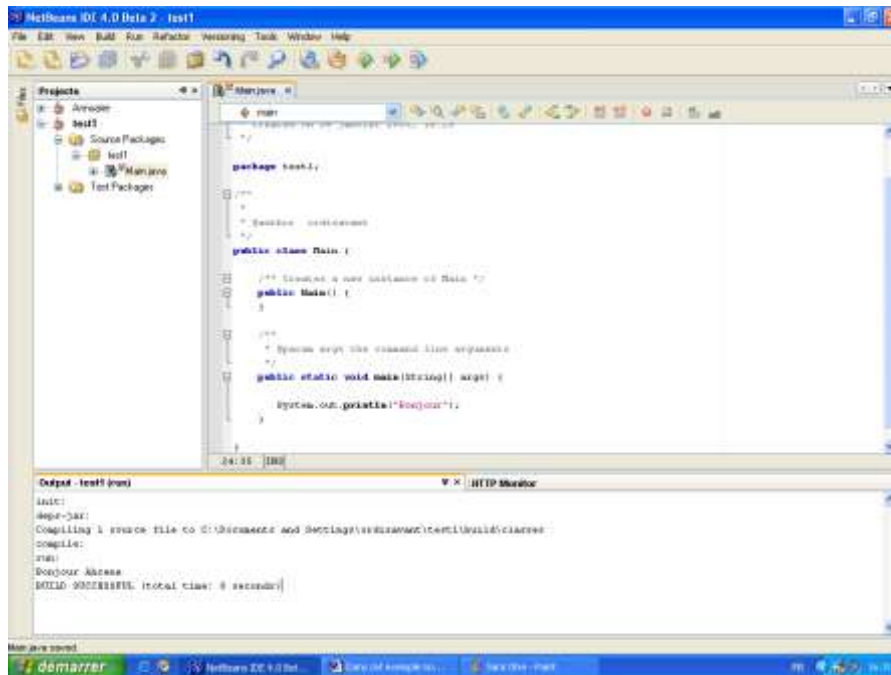
Par Ahcène BOUNCEUR

## INTRODUCTION

Dans cet exemple nous allons vous montrer comment peut-on créer une simple application Java en utilisant l'IDE NetBeans. Cette application doit tout simplement afficher la chaîne de caractères "Bonjour".

### Etape 1 : Création d'un nouveau projet

Lancez NetBeans.



D'abord nous devons créer un projet Java dans lequel se trouveront nos classes. Pour ce faire, allez dans le menu Fichier → Nouveau Projet (ou File → New Project), voir Figure 1.

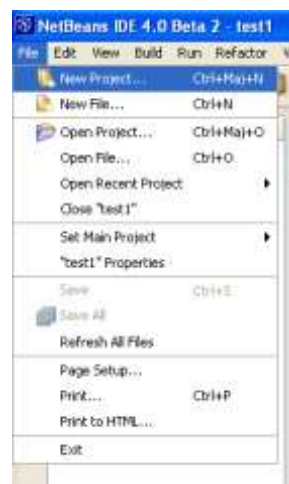


Figure 1

Puis une fenêtre s'affiche (voir Figure 2), dans cette fenêtre spécifiez le type du projet à créer. Vous choisissez donc dans catégories (à gauche) le type standard, puis dans Projects (à droite) choisissez Java Application.

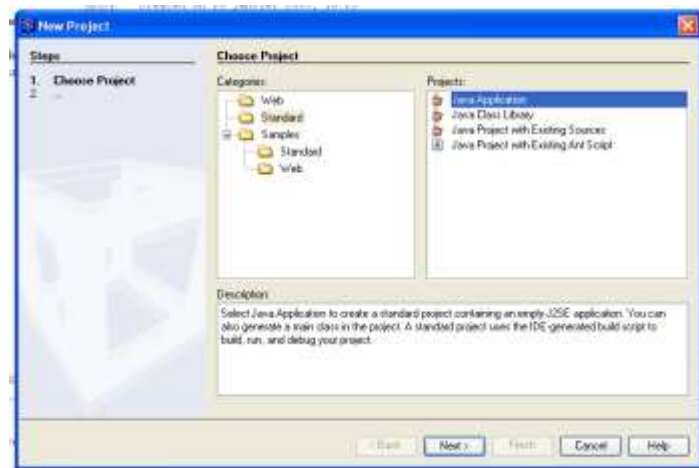


Figure 2

Puis cliquez sur le bouton **Next**, une autre fenêtre s'affiche (voir Figure 3), dans laquelle il faut entrer le nom du projet dans la partie **Project Name**, et faites entrer le nom "test\_bonjour" et contrairement à ECLIPSE, dans NetBeans la classe main peut être créée au même temps que le projet si la case **Create Main Class** est cochée. Cochez donc cette case, puis faites entrer dans le champ correspondant test\_bonjour.classe\_bonjour pour nommer la classe main classe\_bonjour.

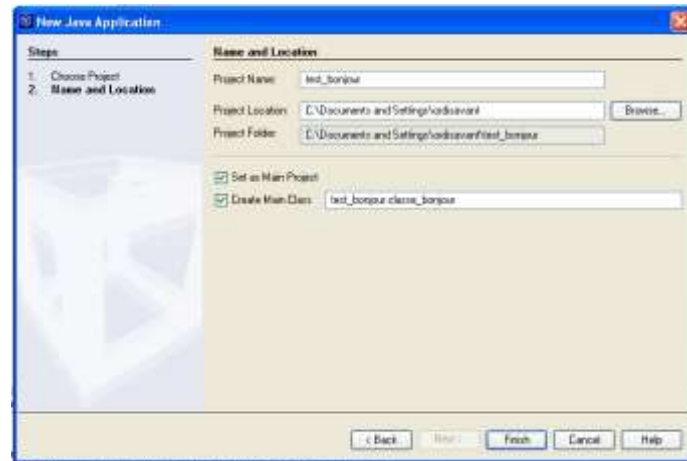
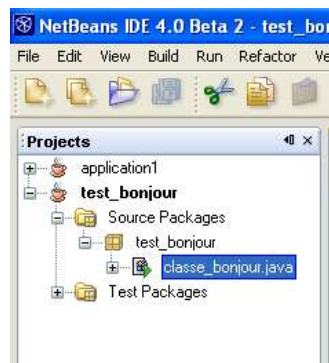


Figure 3

Puis cliquez sur le bouton **Finish**. Le projet est créé (voir Figure 4-a) ainsi que la classe main classe\_bonjour. Si vous ne voyez pas les projets comme dans la Figure 4-a alors cliquez dans le menu sur Window puis sur Projects, voir Figure 4-b.



- a -



- b -

Figure 4

**Question 1 :** Que signifie un projet créé, au niveau du disque-dur ?

**Exercice 1 :** Comment peut on connaître le chemin où se trouve le projet ?

## Étape 2 : Création de la classe principale

La classe principale avec NetBeans peut être créée au même temps que le projet. Nous l'avons déjà créée dans l'étape 1. Si cette classe n'est pas déjà créée alors suivez les mêmes instructions pour créer un package (voir l'étape 5).

Un **package** est un répertoire créé dans votre projet, et la Figure 5 montre bien que la classe `classe_bonjour` se trouve dans un package nommé (`test_bonjour`) ; vérifiez dans votre disque si un tel répertoire a été créé.

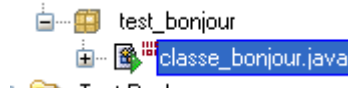


Figure 5

**Question 2 :** Un tel répertoire existe-il ou non ? Où est ce que se trouve donc le fichier source `classe_bonjour.java` dans votre disque ? Conclusion ?

## Étape 3 : Écriture du contenu de la classe `classe_bonjour`

Le contenu du fichier `classe_bonjour.jar` est affiché dans une fenêtre texte (voir Figure 6).

```

/*
 * classe_bonjour.java
 *
 * Created on 16 janvier 2008, 18:14
 */
package test_bonjour;

/**
 *
 * @author erdisavant
 */
public class classe_bonjour {
    /** Creates a new instance of classe_bonjour */
    public classe_bonjour() {
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}

```

Figure 6

Nous allons donc modifier le contenu du code source `classe_bonjour.java` en ajoutant un code permettant d'afficher "Bonjour" à l'intérieur de la méthode `main` (voir Figure 7).

```
public static void main(String[] args) {
    System.out.println("Bonjour");
}
```

Figure 7

Pour compiler ce code, cliquez dans le menu sur **Build** puis sur **Build main Project** (voir Figure 8). Si votre projet n'est pas le projet principal (main project) alors cliquez sur le projet avec le bouton droit de la souris puis cliquez sur **Set Main Project** puis compilez comme c'est décrit précédemment. Et pour l'exécuter soit cliquez sur le bouton **Run Main Project** (voir Figure 9) soit en cliquant dans le menu sur **Run** puis sur **Run Author** puis sur **Test "test\_bonjour"** (voir Figure 10).

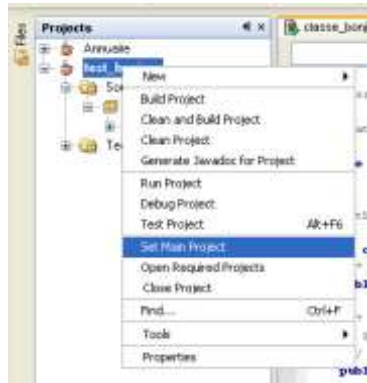


Figure 8

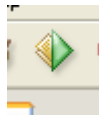


Figure 9

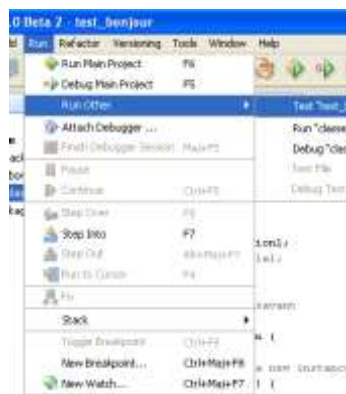


Figure 10

**Question 3 :** Observez le résultat de l'exécution puis donnez le nom de la fenêtre qui affiche le résultat et ainsi que le résultat.

## Étape 4 : La classe classe\_bonjour sous forme d'une applet

**Exercice 2 :** Comment créer une applet qui affiche Bonjour avec NetBeans ?

**Question 4 :** Le résultat est-il le même que celui d'une application ? Où est-ce que le mot "Bonjour" est affiché ?

**Exercice 3 :** Créez une page-web qui utilise cette applet.

## Étape 5 : Création et utilisation d'un package

Pour créer un package, cliquez sur le projet avec le bouton droit puis sur **nouveau** puis sur **Java Package...** (voir Figure 12).





Figure 12

Une fenêtre s'affiche.

**Question 5 :** Faite entrer, dans cette fenêtre, comme nom le mot “package”, est il possible ? Sinon, entrer le nom “package1”

**Question 6 :** Pourquoi le mot “package” ne peut pas être un nom de notre package ?

Le résultat est présenté dans la Figure 13.

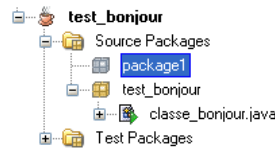


Figure 13

Créez une classe appelée “classe1” dans le package “package1”.

**Question 7 :** Quelles sont les démarches à suivre pour créer cette classe ?

Créez dans la classe classe1 un attribut appelé “s” de type `String` et un constructeur qui initialise la variable `s` à la valeur “Salut” et une méthode appelée `valeur_de_s` qui renvoie la valeur de `s`.

**Question 8 :** Que faut-il écrire ?

Modifiez le code de la classe `main` en affichant à la place de “bonjour” la valeur de l'attribut `s` d'un objet de type `classe1`.

**Question 9 :** Que faut-il écrire ? Exécutez, et dire ce qui est affiché.

Vous avez sûrement ajouté la commande `import package1.classe1`, sinon erreur.

## Étape 6 : CLASSPATH

### 1 – Appel d'une classe dans un package local

**Question 10 :** La commande `import package1.classe1` est elle toujours nécessaire ?

**Question 11 :** Enlevez cette commande (ou ligne) puis exécutez, c'est quoi le résultat ?

### 2 – Appel d'une classe dans un JAR

Créez un répertoire dans votre disque et appelez le `mes_jars`, puis télécharger le fichier `classe_ext.jar` sur l'adresse suivante : <http://>, et finalement placez ce fichier dans le répertoire `mes_jars` que vous venez de créer.

**Question 12 :** Que faut-il faire pour pouvoir utiliser les différentes classes de ce JAR ? Quelles sont alors les classes que contient ce JAR (sans ouvrir le fichier JAR) ?

**Question 13 :** Ce JAR contient une classe appelée `classe_plus`, quels sont les attributs de cette classe ? Ils sont de quels types ?

**Question 14 :** Quelles sont les différents constructeurs et les différentes méthodes de cette classe ? Les méthodes sont de quels types ?

**Question 15 :** La classe `classe_plus` contient uniquement un seul attribut de type `String`. Utilisez votre applet pour afficher la valeur de cet attribut. Quelle est donc la valeur de cet attribut ?

### 3 – Créer un JAR

**Exercice 4 :** Décrivez toutes les étapes à suivre pour créer un JAR.

**Exercice 5 :** Créez un JAR qui contiendra tout le projet que vous venez de créer. Puis créez un nouveau projet qu'il faut appeler `projet_jar`. Décrivez les démarches à suivre pour exécuter le code qui se trouve dans ce JAR.

### Étape 8 : Le Javadoc

Il existe deux types de commentaires dans les codes Java. Le premier type représente les commentaires classiques qui sont sous la forme suivante :

```
/*  
Ligne 1 de mon commentaire,  
Ligne 2 de mon commentaire,  
...  
*/
```

Ce type de commentaire est utile pour le programmeur. Le deuxième type a le même principe que le premier mais il est utile pour la génération du Javadoc. Il s'écrit comme suit :

```
/**  
Ligne 1 de mon commentaire,  
Ligne 2 de mon commentaire,  
...  
*/
```

Chaque commentaire de type 2 doit définir la méthode le suivant.

Ajouter à votre programme tous les commentaires de type 2 définissant toutes les méthodes et attributs.

**Question 16 :** Comment générer un Javadoc ?