Log For Java

- **Comprendre la problématique des traces applicatives**
- **Log4J**
  - Connaître les avantages de Log4J
  - Utiliser Log4j
  - Configurer Log4J

OBJECTIFS PEDAGOGIQUES

Log For Java

http://logging.apache.org/log4j/

But

- **Être une solution «propre» de logging:**
  - **Remplace les System.err ou System.out**
- **Avantages d'une API de logging:**
  - Activer/Désactiver **la capture d'information**
  - **Produire des rapports** formatés **de logs**
  - **Catégoriser les logs, i.e.** sélectionner **ce que l'on veut logger.**
  - Définir **des** niveaux de logs
  - Rediriger les traces **dans une console et/ou vers un fichier, une socket sur le réseau, un manager SNMP...**
- Ne jamais toucher au code!

Configuration

- Configuration de Log4J
  - Créer et configurer un fichier

log4j.properties ou log4j.xml

- La configuration peut être rechargée par l'application
- Note:
  - La config XML prime sur les properties

Niveaux de logs

Logger Level	Will Output Messages Of Level				
	DEBUG	INFO	WARN	ERROR	FATAL
DEBUG					
INFO					
WARN					
ERROR					
FATAL					
ALL					
OFF					

Appenders

- Sortie choisie des logs :
  - Console
  - Fichier
  - Socket
  - JMS
  - …

Layouts

➔ Formattage des logs

- Simple layout
  - produit des logs très simples.
- PatternLayout
  - produit des logs suivant un pattern données.
- HTML layout
- XMLLayout

Catégories

➔ Permet de logger suivant un niveau (i.e. DEBUG) une classe ou un package et le reste suivant un autre niveau (i.e. INFO)

- **Pour le package com.garage on trace les logs à partir du niveau INFO dans l'appender htmlfile**
- **Pour le package com.prefecture on trace les logs à partir du niveau DEBUG dans l'appender console**

```
<categoryname="com.garage">
        <priorityvalue="INFO" />
        <appender-refref="htmlfile"/>
</category>
<categoryname="com.prefecture">
        <priorityvalue="DEBUG" />
        <appender-efref="console" />
</category>
```

Codage d'un log

➔ Définition du logger:

```
import org.apache.log4j.Logger;
…
private static final Logger log = Logger.getLogger(MyClass.class);
```

➔ Écriture d'un log:

```
log.debug("myText");
log.info("myText");
log.error("myText");
log.fatal("myText");
```

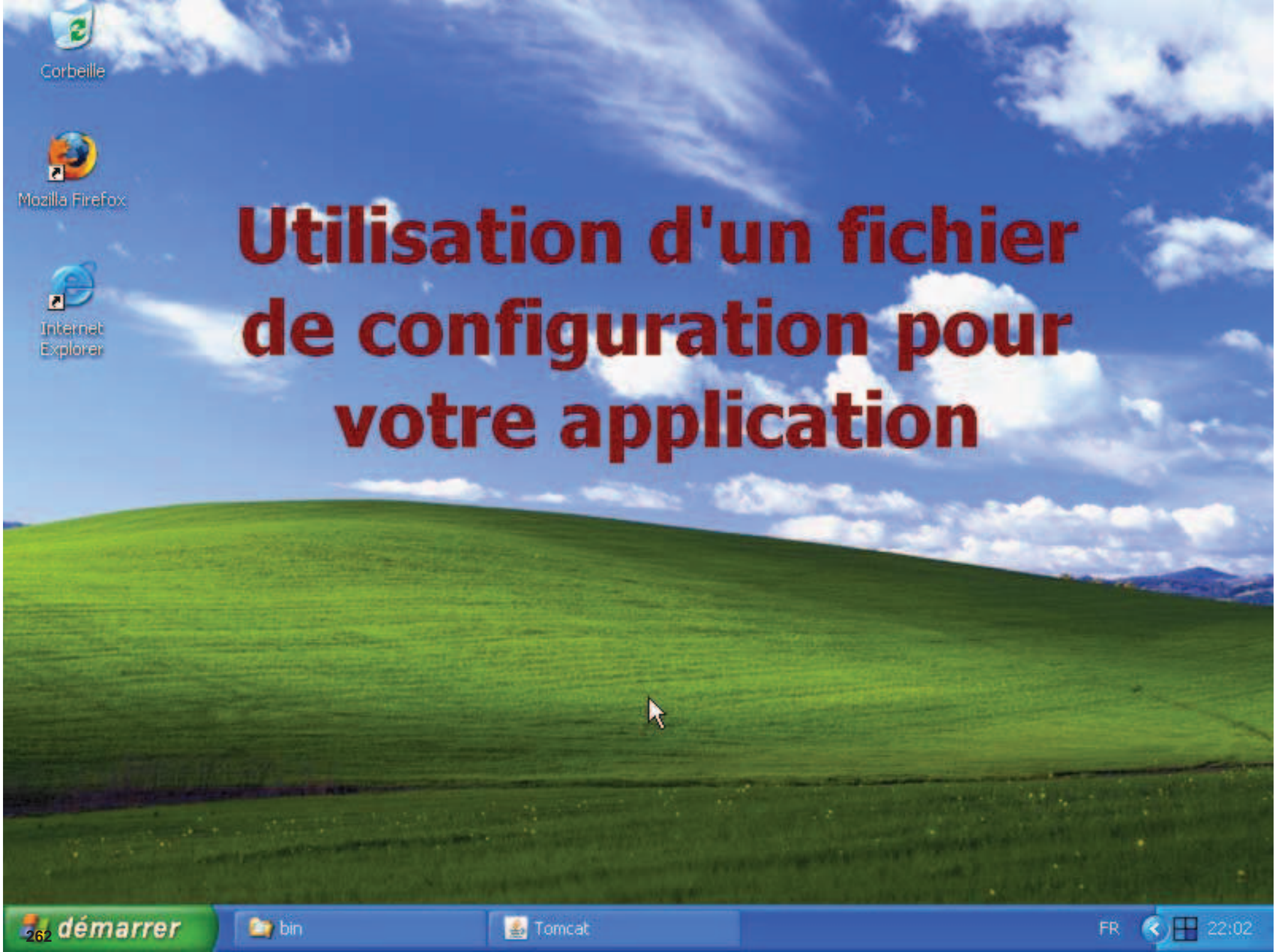
➔ Utiliser StringBuffer au lieu la concaténation avec le symbole +:

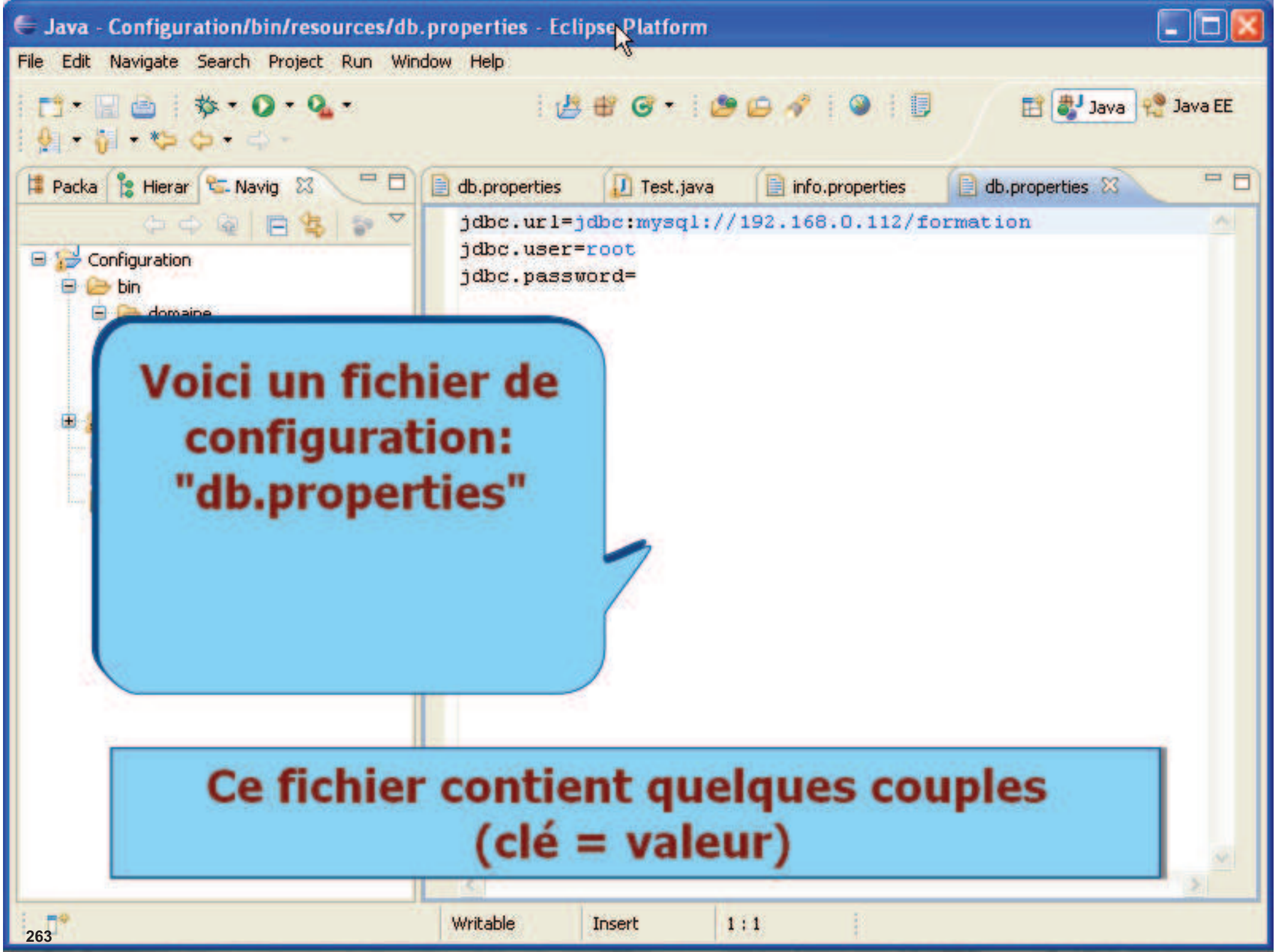
```
StringBuffer str = new StringBuffer("début");
str.append(" …");
str.append(" fin");
```

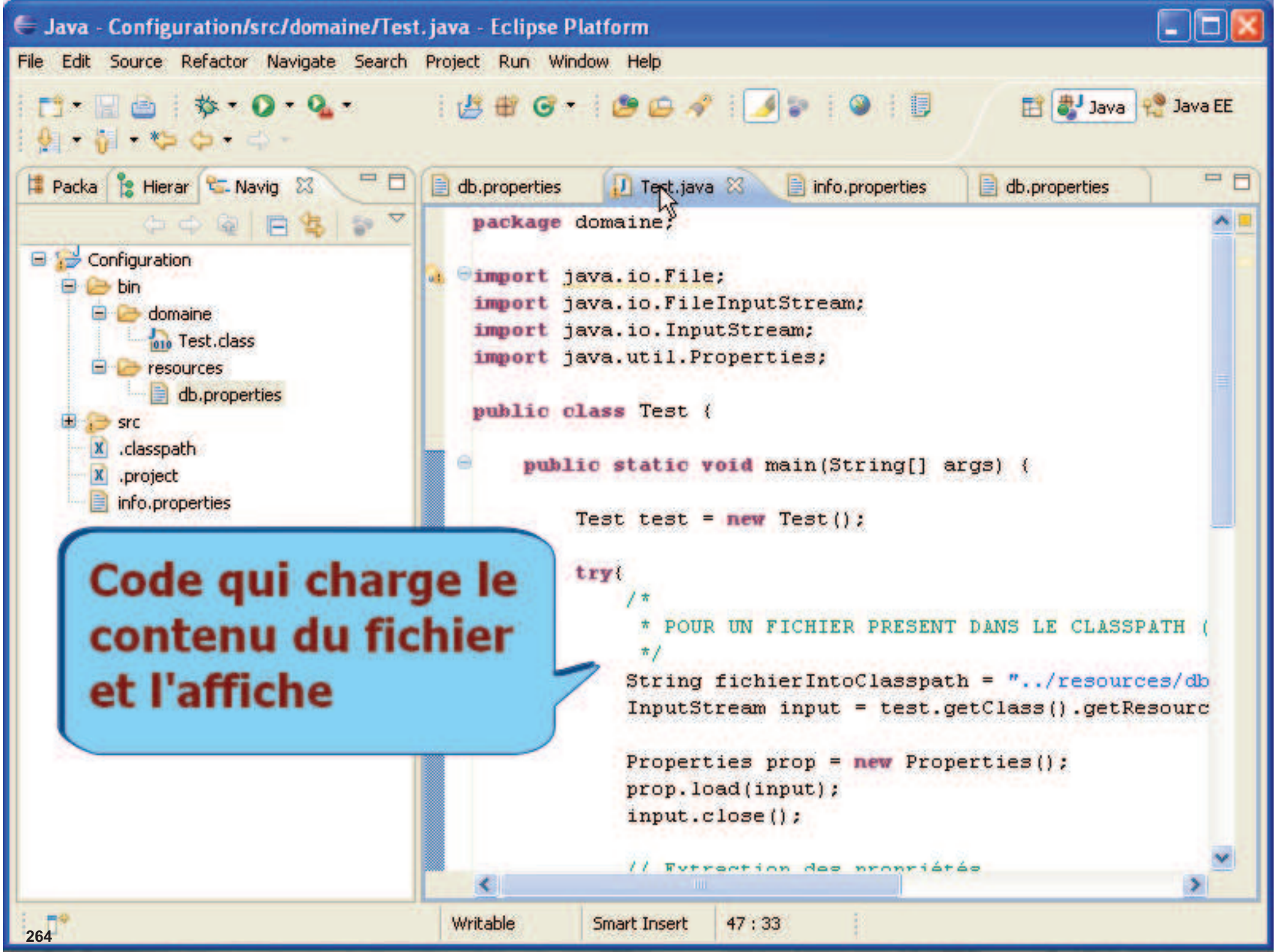
Log4J

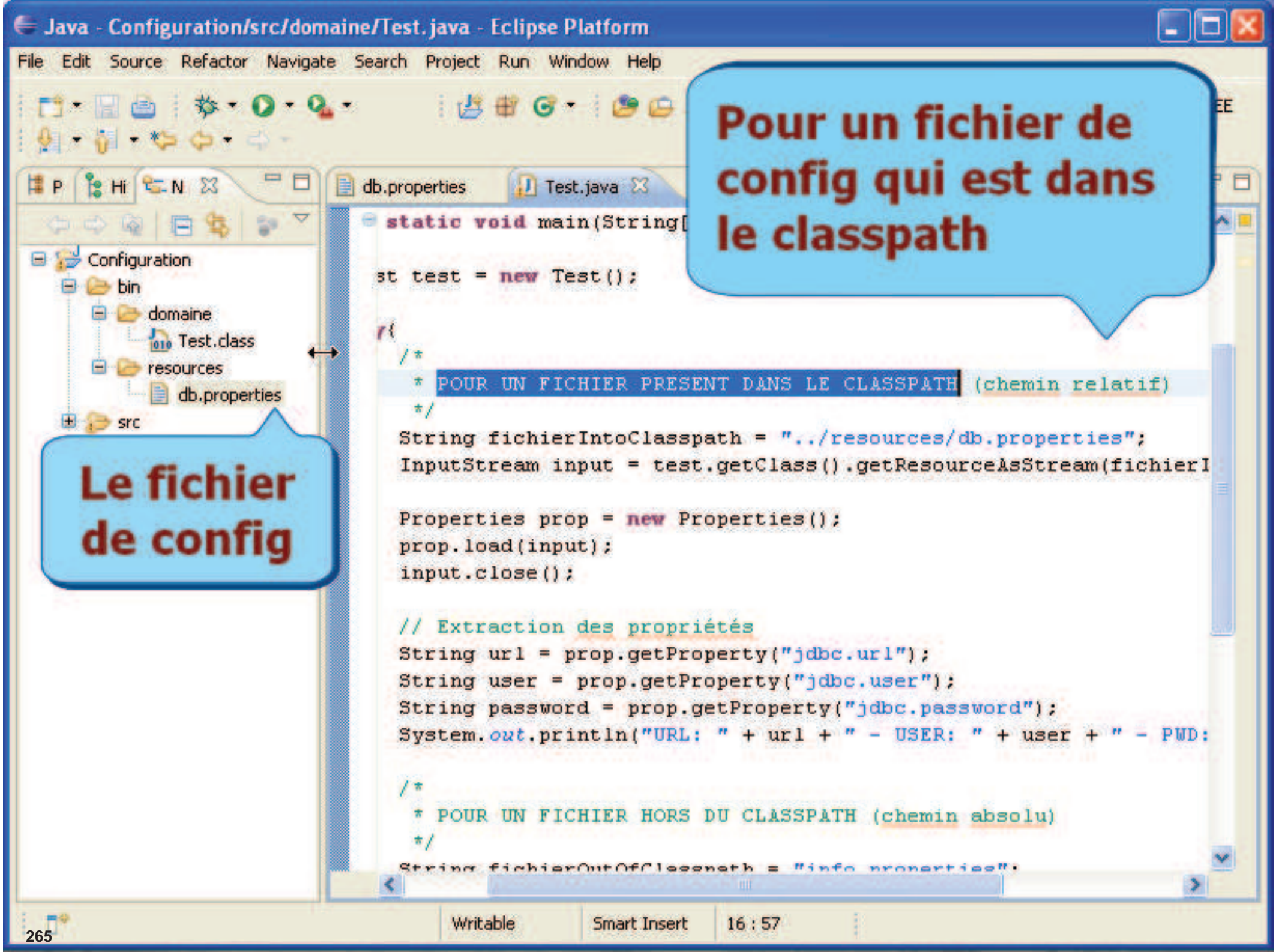
➔ **Solution «propre» de gestion des traces applicatives:**
- **Remplace les System.err ou System.out**

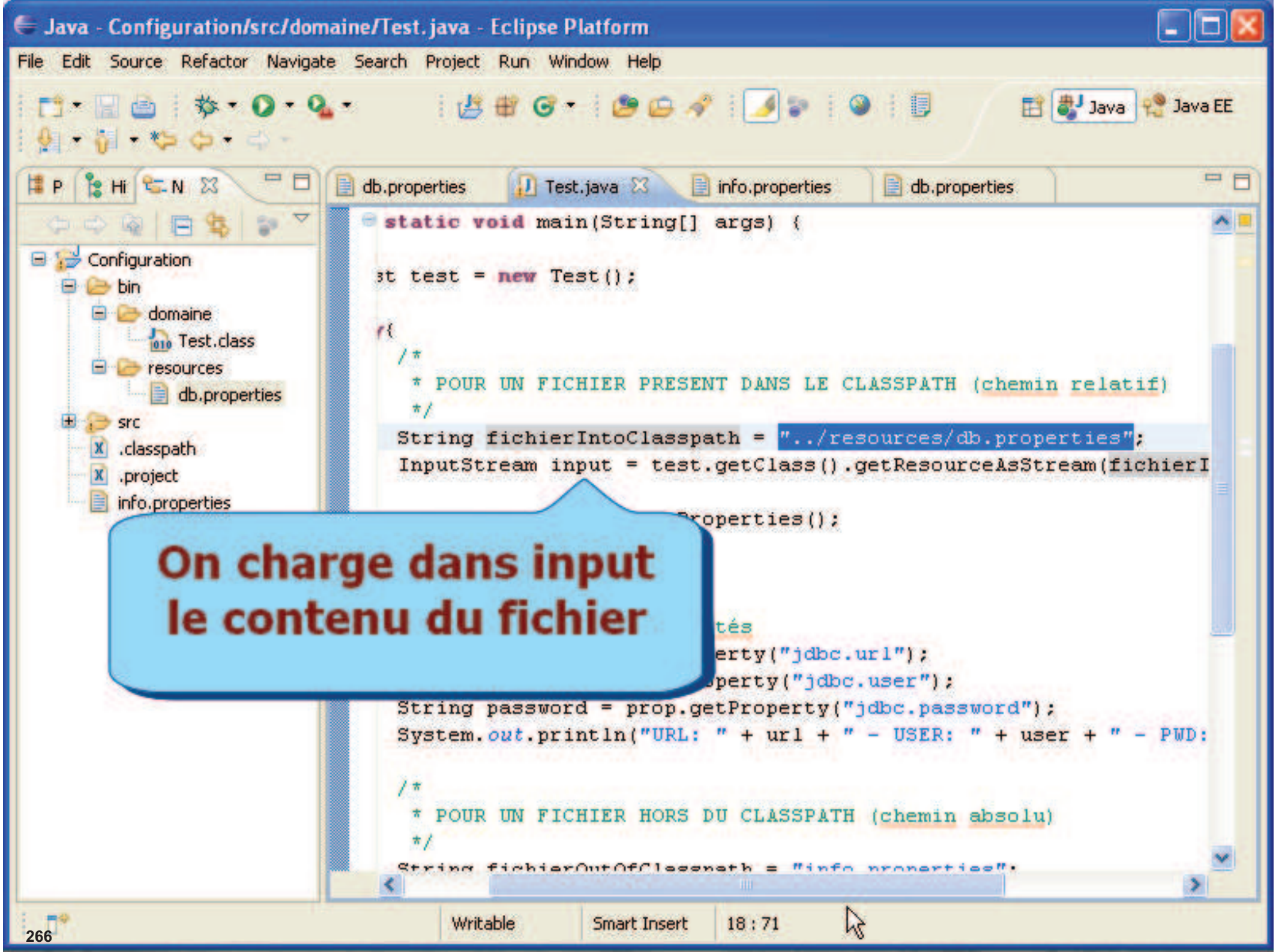
➔ **Avantages d'une API de logging:**
- **Activer/Désactiver** la génération de logs
- Produire des logs **formatés**
- **Catégoriser les logs**, i.e. sélectionner ce que l'on veut logger.
- Définir des **niveaux de logs**
- **Rediriger les traces** dans une console et/ou vers un fichier, une socket sur le réseau, un manager SNMP...
- Pas de perte en **performances**
- Pas besoin de **recompiler**, ni d'**arrêter le service**

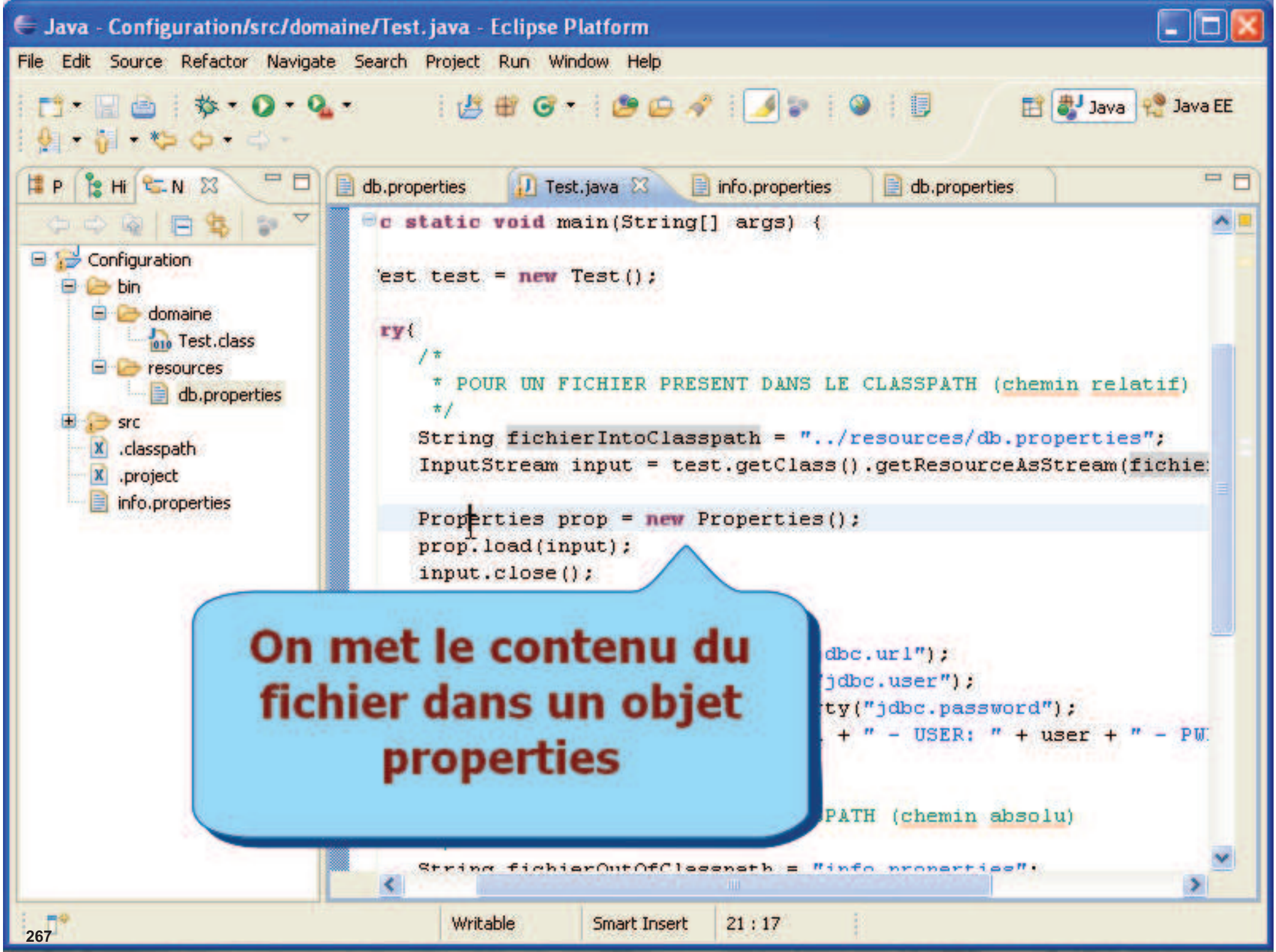

Corbeille
Mozilla Firefox
Internet Explorer
Utilisation d'un fichier de configuration pour votre application
démarrer
bin
Tomcat
FR
22:02

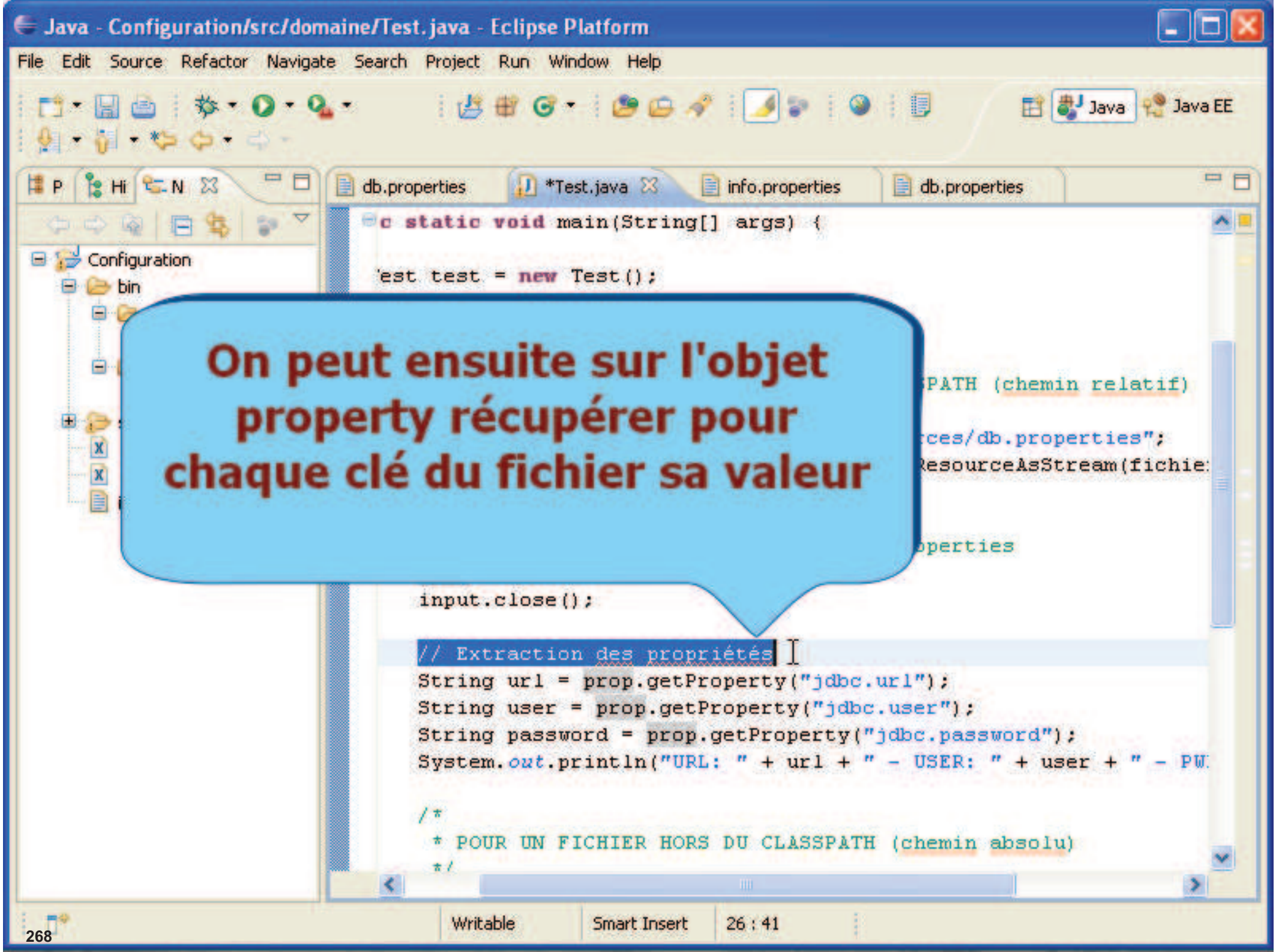

Java - Configuration/bin/resources/db.properties - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Java
Java EE
Packa
Hierar
Navig
db.properties
Test.java
info.properties
db.properties
jdbc.url=jdbc:mysql://192.168.0.112/formation
jdbc.user=root
jdbc.password=
Configuration
bin
Voici un fichier de configuration: "db.properties"
Ce fichier contient quelques couples (clé = valeur)
Writable
Insert
1 : 1

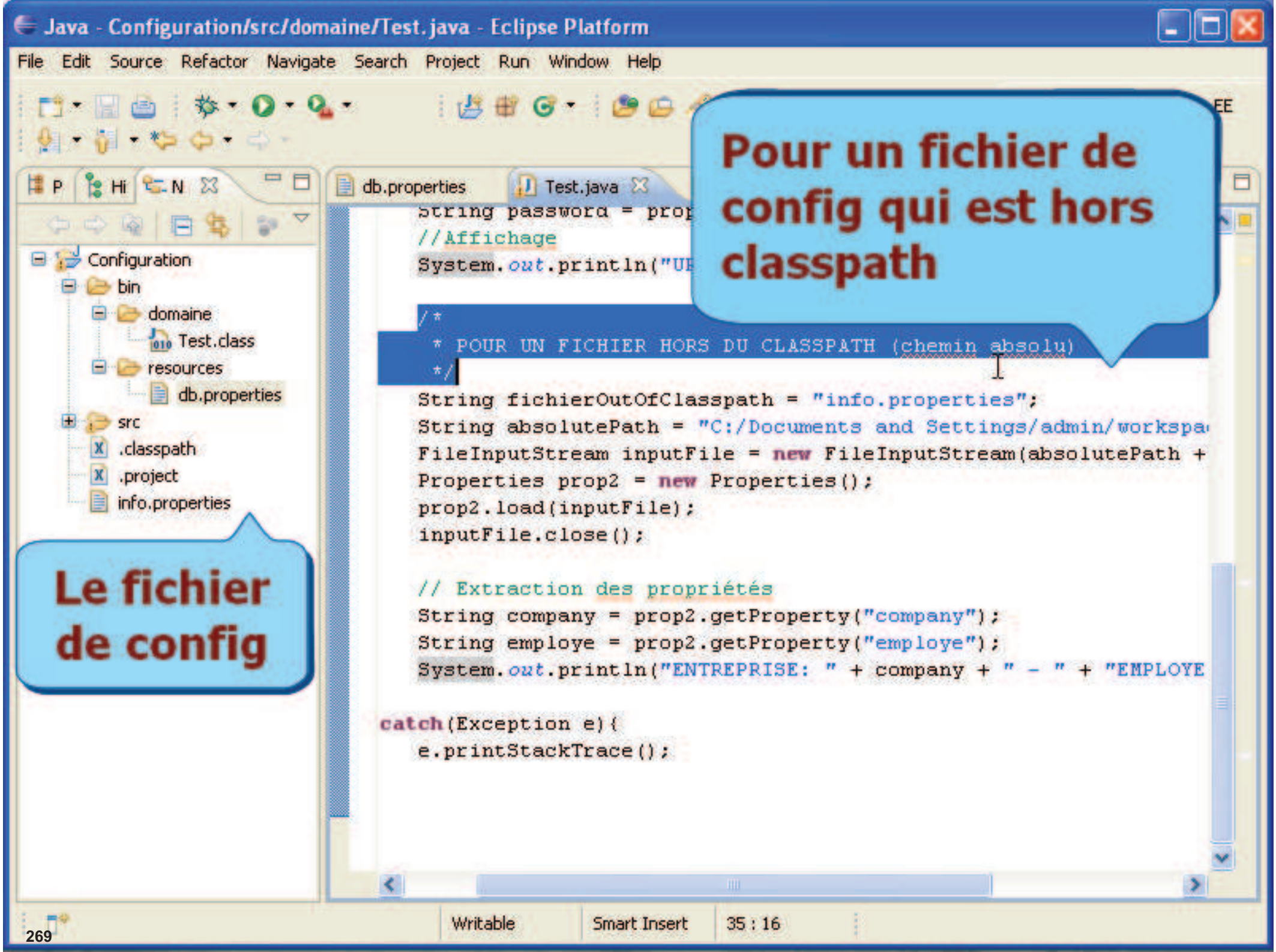

Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Packa
Hierar
Navig
db.properties
Test.java
info.properties
db.properties
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
package domaine;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Properties;
public class Test {
public static void main(String[] args) {
Test test = new Test();
try{
/*
* POUR UN FICHIER PRESENT DANS LE CLASSPATH (
*/
String fichierIntoClasspath = "../resources/db
InputStream input = test.getClass().getResourc
Properties prop = new Properties();
prop.load(input);
input.close();
Code qui charge le contenu du fichier et l'affiche
Writable
Smart Insert
47 : 33

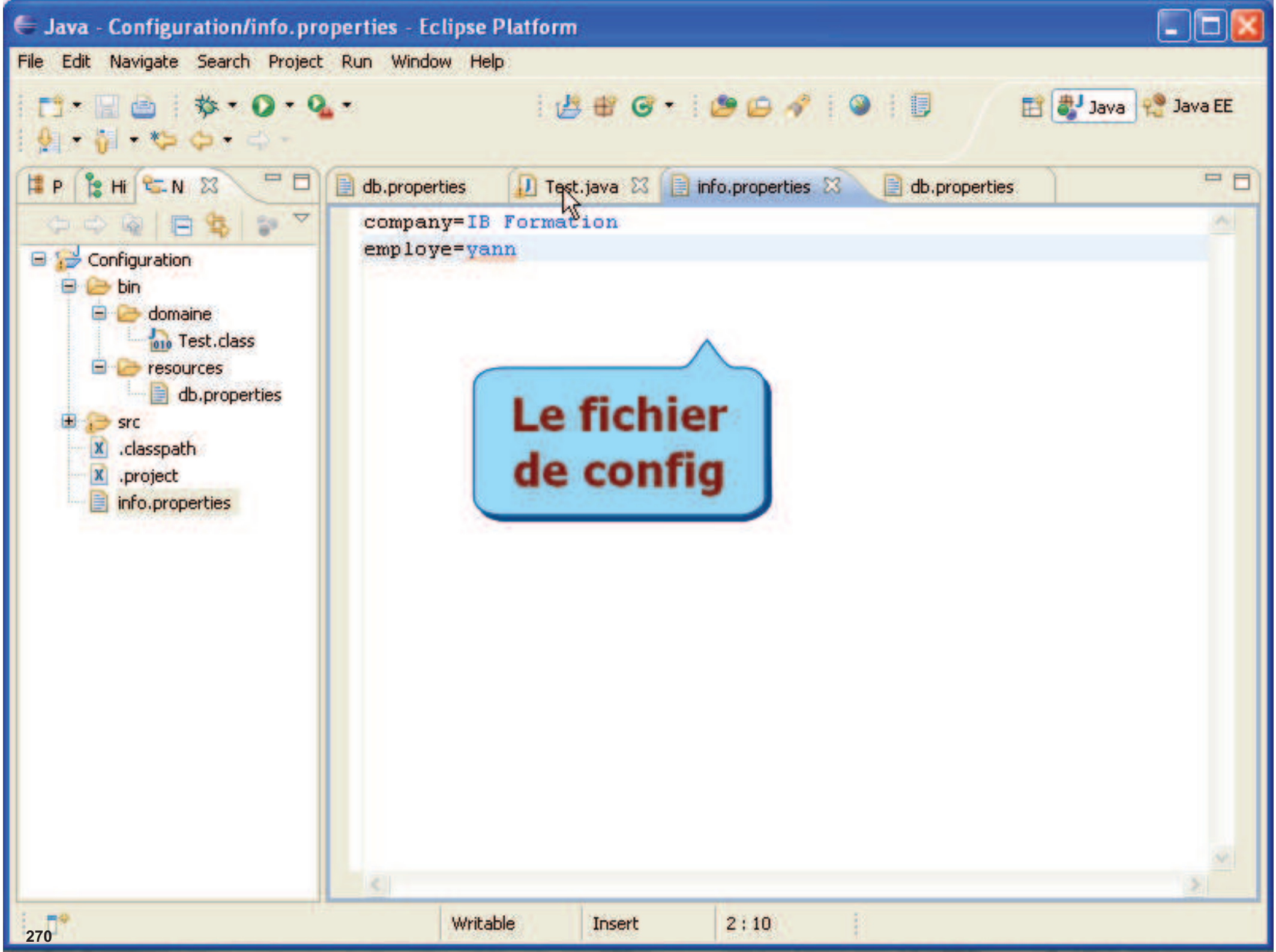

Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Pour un fichier de config qui est dans le classpath
Configuration
bin
domaine
Test.class
resources
db.properties
src
Le fichier de config
db.properties
Test.java
static void main(String[
st test = new Test();
r{
/*
* POUR UN FICHIER PRESENT DANS LE CLASSPATH (chemin relatif)
*/
String fichierIntoClasspath = "../resources/db.properties";
InputStream input = test.getClass().getResourceAsStream(fichierI
Properties prop = new Properties();
prop.load(input);
input.close();
// Extraction des propriétés
String url = prop.getProperty("jdbc.url");
String user = prop.getProperty("jdbc.user");
String password = prop.getProperty("jdbc.password");
System.out.println("URL: " + url + " - USER: " + user + " - PWD:
/*
* POUR UN FICHIER HORS DU CLASSPATH (chemin absolu)
*/
Writable
Smart Insert
16 : 57

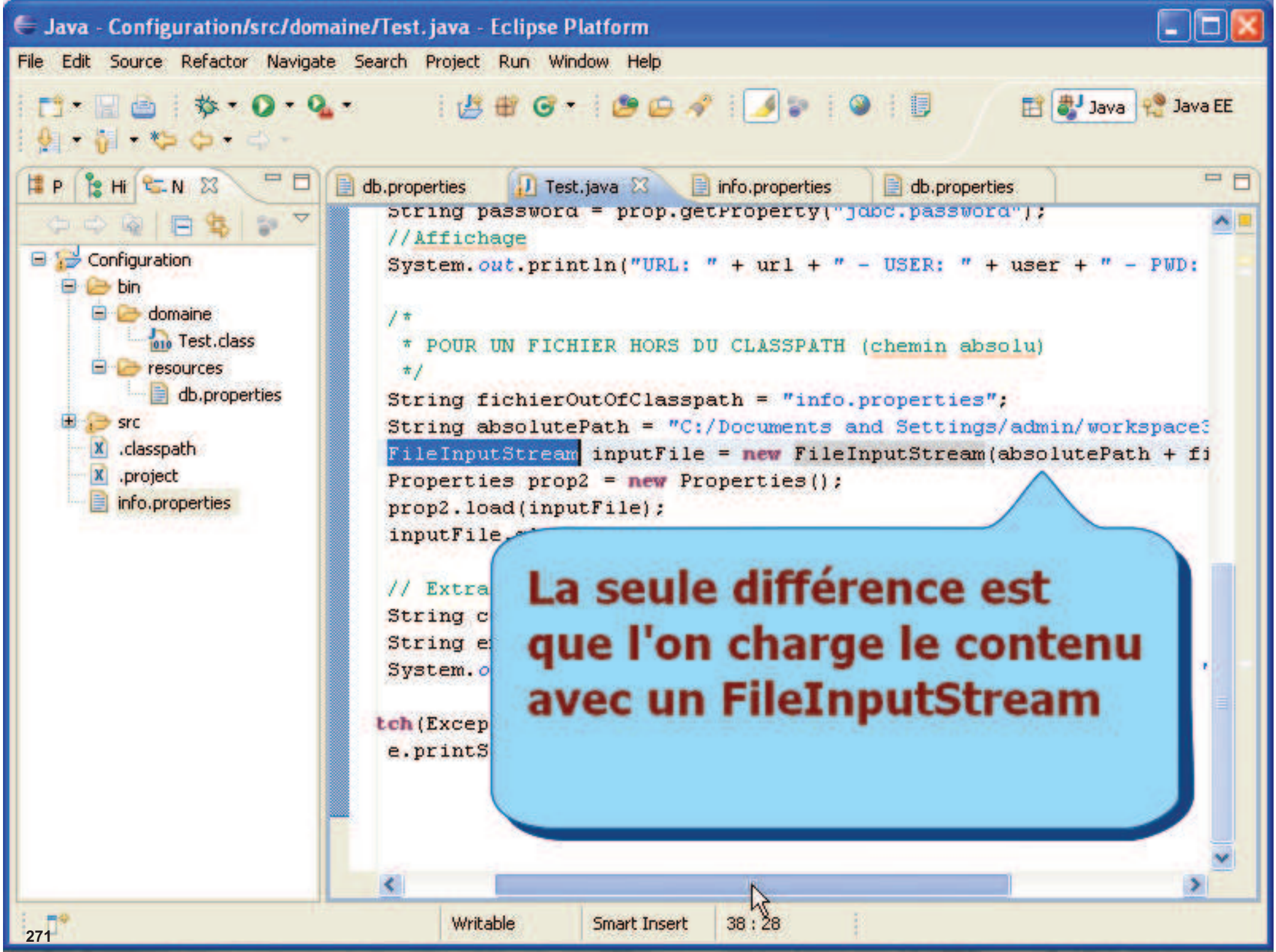

Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
db.properties
Test.java
info.properties
db.properties
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
static void main(String[] args) {
st test = new Test();
{
/*
* POUR UN FICHIER PRESENT DANS LE CLASSPATH (chemin relatif)
*/
String fichierIntoClasspath = "../resources/db.properties";
InputStream input = test.getClass().getResourceAsStream(fichierI
roperties();
tés
erty("jdbc.url");
perty("jdbc.user");
String password = prop.getProperty("jdbc.password");
System.out.println("URL: " + url + " - USER: " + user + " - PWD:
/*
* POUR UN FICHIER HORS DU CLASSPATH (chemin absolu)
*/
On charge dans input
le contenu du fichier
Writable
Smart Insert
18 : 71

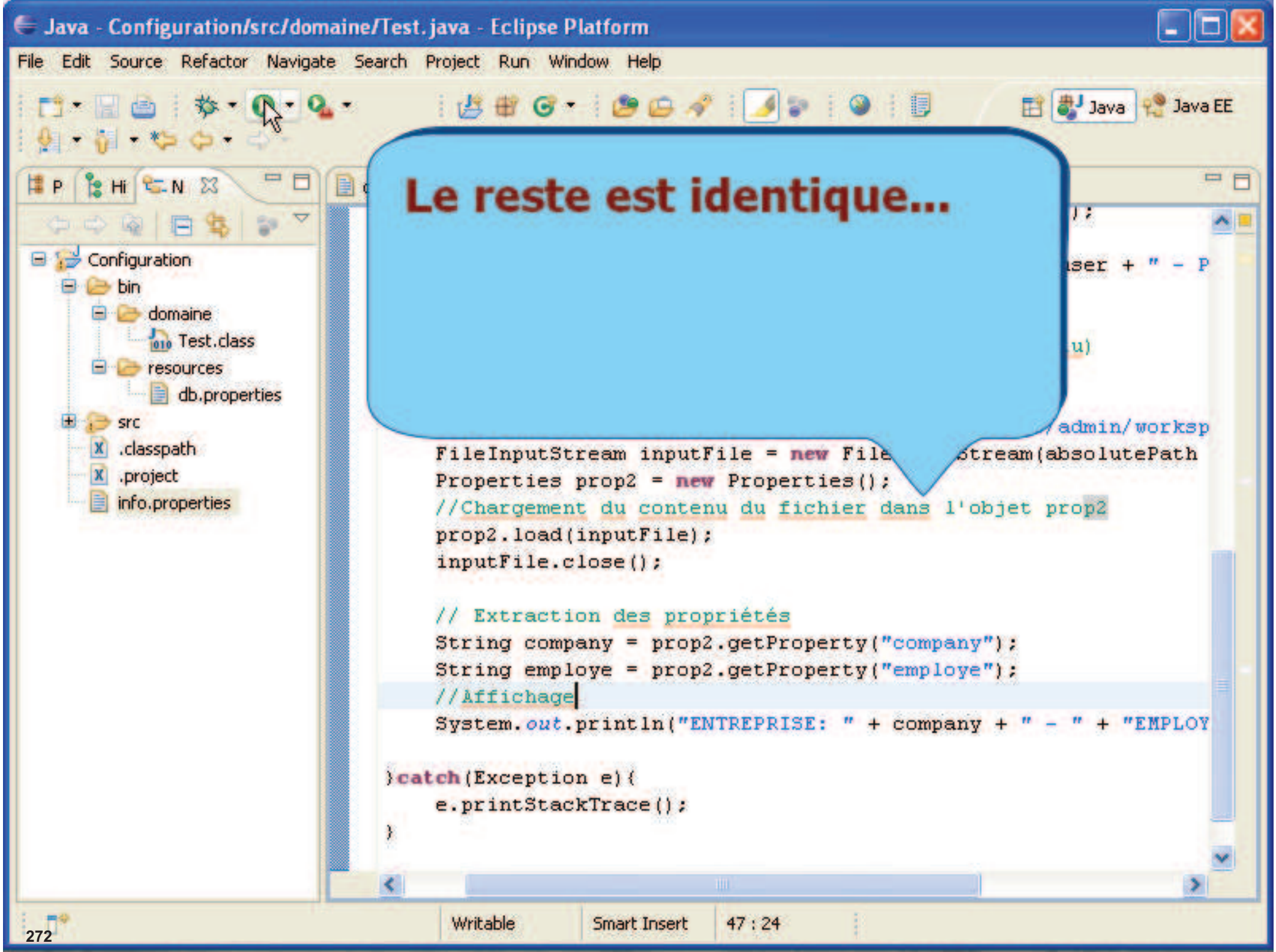

Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
db.properties
Test.java
info.properties
db.properties
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
c static void main(String[] args) {
'est test = new Test();
ry{
/*
* POUR UN FICHIER PRESENT DANS LE CLASSPATH (chemin relatif)
*/
String fichierIntoClasspath = "../resources/db.properties";
InputStream input = test.getClass().getResourceAsStream(fichie
Properties prop = new Properties();
prop.load(input);
input.close();
dbc.url");
jdbc.user");
ty("jdbc.password");
+ " - USER: " + user + " - PW
PATH (chemin absolu)
On met le contenu du fichier dans un objet properties
Writable
Smart Insert
21 : 17

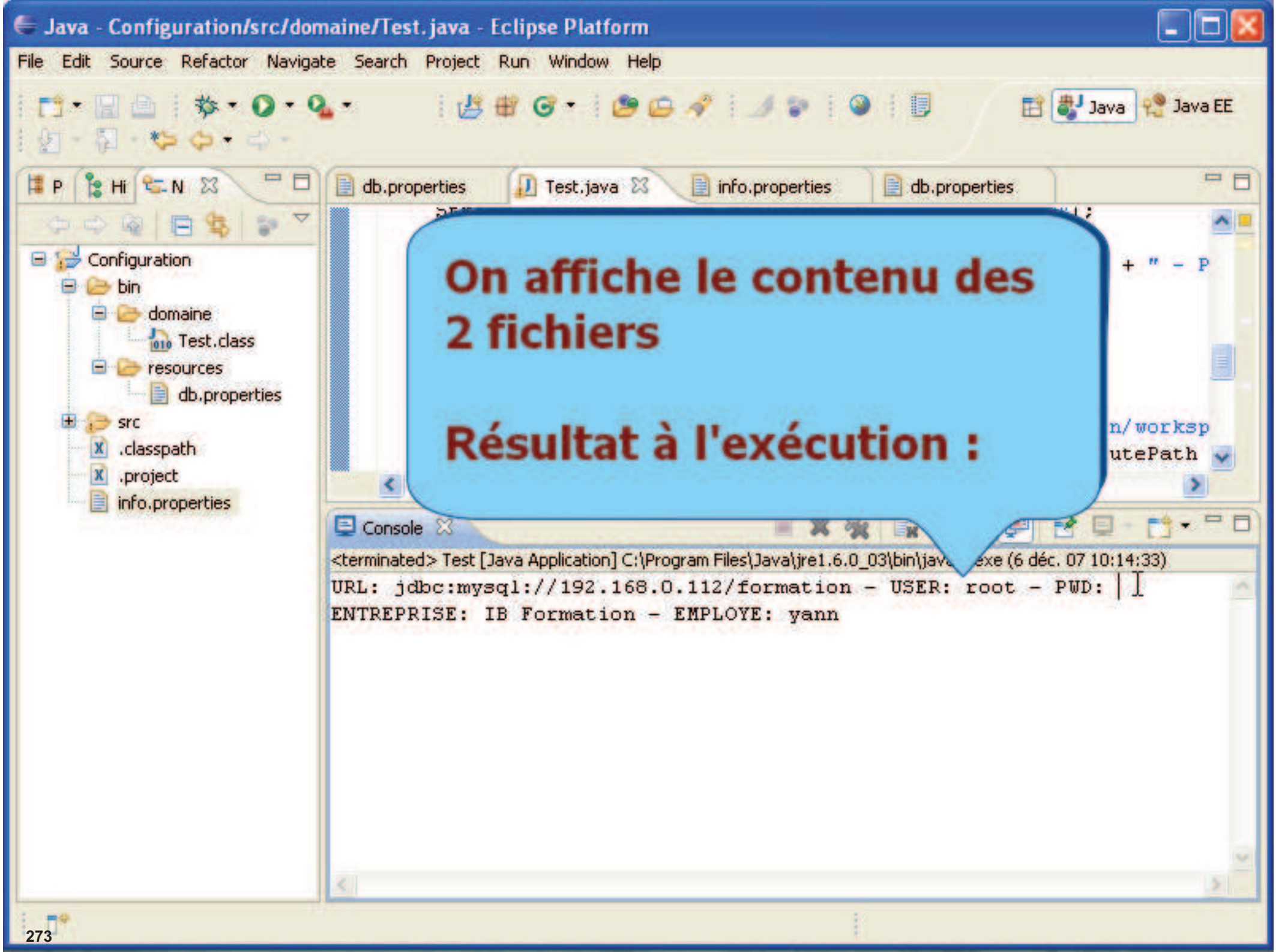

Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
db.properties
*Test.java
info.properties
db.properties
Configuration
bin
On peut ensuite sur l'objet property récupérer pour chaque clé du fichier sa valeur
c static void main(String[] args) {
est test = new Test();
PATH (chemin relatif)
ces/db.properties";
esourceAsStream(fichie
operties
input.close();
// Extraction des propriétés
String url = prop.getProperty("jdbc.url");
String user = prop.getProperty("jdbc.user");
String password = prop.getProperty("jdbc.password");
System.out.println("URL: " + url + " - USER: " + user + " - PW
/*
* POUR UN FICHIER HORS DU CLASSPATH (chemin absolu)
*/
Writable
Smart Insert
26 : 41


Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
db.properties
Test.java
//Affichage
System.out.println("UR
/*
* POUR UN FICHIER HORS DU CLASSPATH (chemin absolu)
*/
String fichierOutOfClasspath = "info.properties";
String absolutePath = "C:/Documents and Settings/admin/workspa
FileInputStream inputFile = new FileInputStream(absolutePath +
Properties prop2 = new Properties();
prop2.load(inputFile);
inputFile.close();
// Extraction des propriétés
String company = prop2.getProperty("company");
String employe = prop2.getProperty("employe");
System.out.println("ENTREPRISE: " + company + " - " + "EMPLOYE
catch(Exception e){
e.printStackTrace();
Pour un fichier de config qui est hors classpath
Le fichier de config
Writable
Smart Insert
35 : 16


Java - Configuration/info.properties - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Java
Java EE
P
Hi
N
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
db.properties
Test.java
info.properties
db.properties
company=IB Formation
employe=yann
Le fichier de config
Writable
Insert
2 : 10


Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
db.properties
Test.java
info.properties
db.properties
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
String password = prop.getProperty("jdbc.password");
//Affichage
System.out.println("URL: " + url + " - USER: " + user + " - PWD:
/*
* POUR UN FICHIER HORS DU CLASSPATH (chemin absolu)
*/
String fichierOutOfClasspath = "info.properties";
String absolutePath = "C:/Documents and Settings/admin/workspace
FileInputStream inputFile = new FileInputStream(absolutePath + fi
Properties prop2 = new Properties();
prop2.load(inputFile);
La seule différence est
que l'on charge le contenu
avec un FileInputStream
Writable
Smart Insert
38 : 28


Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Le reste est identique...
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
FileInputStream inputFile = new File
Properties prop2 = new Properties();
//Chargement du contenu du fichier dans l'objet prop2
prop2.load(inputFile);
inputFile.close();
// Extraction des propriétés
String company = prop2.getProperty("company");
String employe = prop2.getProperty("employe");
//Affichage
System.out.println("ENTREPRISE: " + company + " - " + "EMPLOY
}catch(Exception e){
e.printStackTrace();
}
Writable
Smart Insert
47 : 24


Java - Configuration/src/domaine/Test.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
db.properties
Test.java
info.properties
db.properties
Configuration
bin
domaine
Test.class
resources
db.properties
src
.classpath
.project
info.properties
On affiche le contenu des 2 fichiers
Résultat à l'exécution :
Console
<terminated> Test [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\java.exe (6 déc. 07 10:14:33)
URL: jdbc:mysql://192.168.0.112/formation - USER: root - PWD:
ENTREPRISE: IB Formation - EMPLOYE: yann

Détecter et Gérer les cas d'erreurs

- Prendre en compte plusieurs langues
- Comment ajouter un langage sans modifier/recompiler l'application
- Comment afficher les nombres, dates, monnaies et pourcentages

OBJECTIFS PEDAGOGIQUES

Internationalisation

- L'internationalisation **est le processus de conception d'une application pour qu'elle puisse être adaptée aux différentes** langues **et** régions **sans aucun changement technique majeur.**

- **Parfois, le terme de l'internationalisation est abrégé comme l'**i18n**, car il ya 18 lettres entre le premier "i" et le dernier "n"**

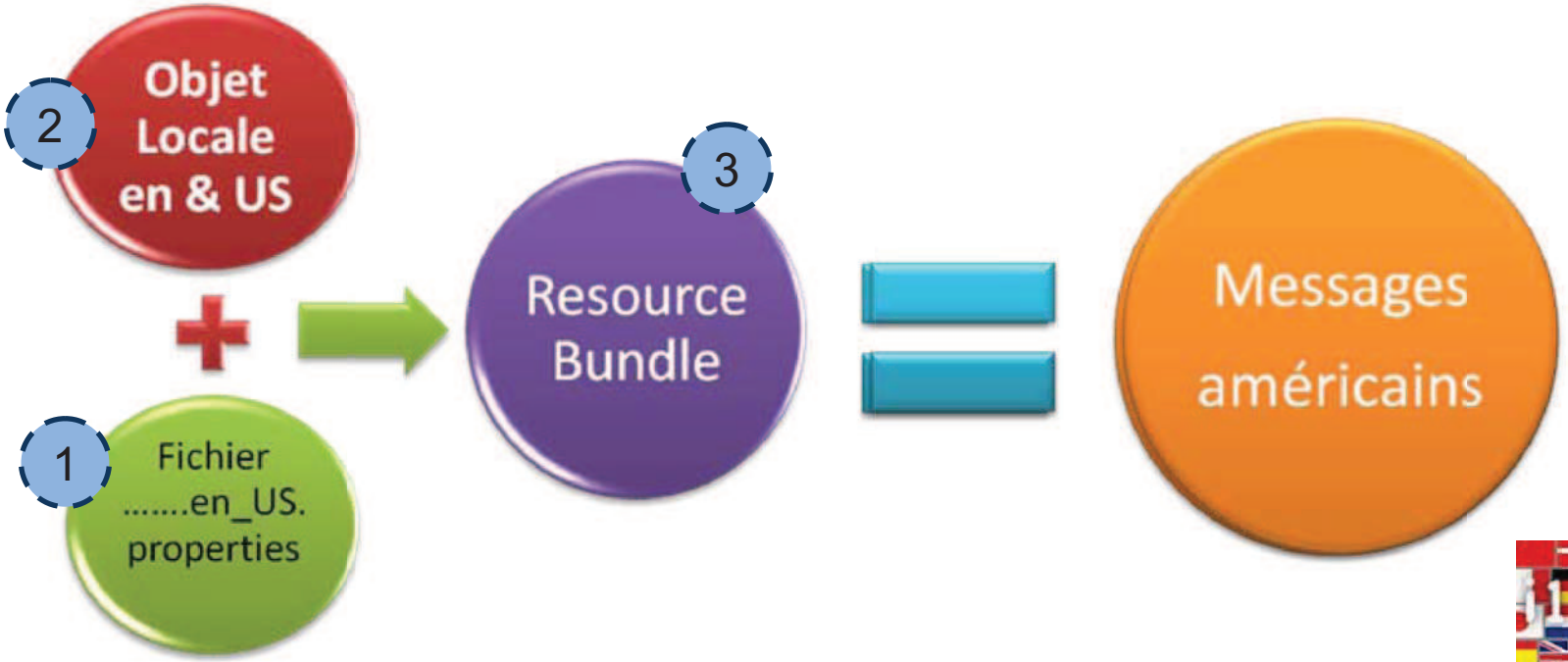
Avantages

➔ Caractéristiques d'une application i18n:

- **Le** même exécutable **est utilisé** dans le monde entier
- **Les éléments, tels que des** messages **de l'interface graphique, les** labels**, les noms des** boutons**, ne sont pas en dur dans le programme. Au contraire, ils sont** stockés en dehors du code source **et de récupérer dynamiquement**
- **Le support de nouvelles langues ne nécessite pas de refonte.**
- **Les données culturelles, comme les** dates **et les** monnaies**, apparaissent dans des** formats **qui sont conformes à l'utilisateur**

A : Pour les messages

- ➔ 1. Créer un fichier properties par langue-région
- ➔ 2. Définir la langue, région avec la classe Locale (en & US)
- ➔ 3. Créer un objet ResourceBundle
- ➔ 4. Récupérer les messages grâce au ResourceBundle



Les étapes : le code

➔ 1. Créer un fichier properties par langue-région

Exemple: monfichier_en_US.properties

Ce fichier plusieurs ligne, chaque ligne contient une clé '=' la valeur :

greetings = Hello

➔ 2. Définir la langue, région avec la classe Locale

```
Locale loc = new Locale ("en", "US");
```

➔ 3. Créer un objet ResourceBundle

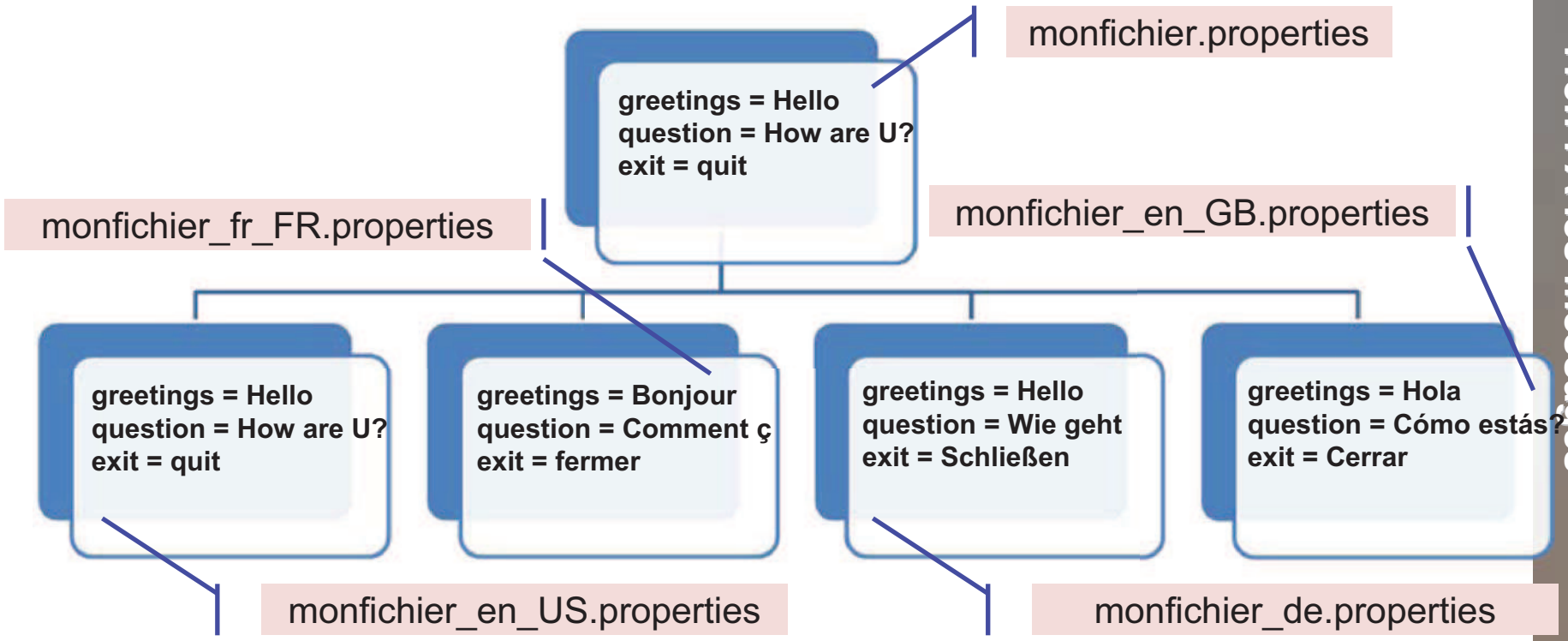
```
ResourceBundle messages;
messages = ResourceBundle.getBundle("monfichier", loc);
```

➔ 4. Récupérer les messages grâce au ResourceBundle

```
 String msg1 = messages. .getString("greetings");
// msg1 vaut "Hello"
```

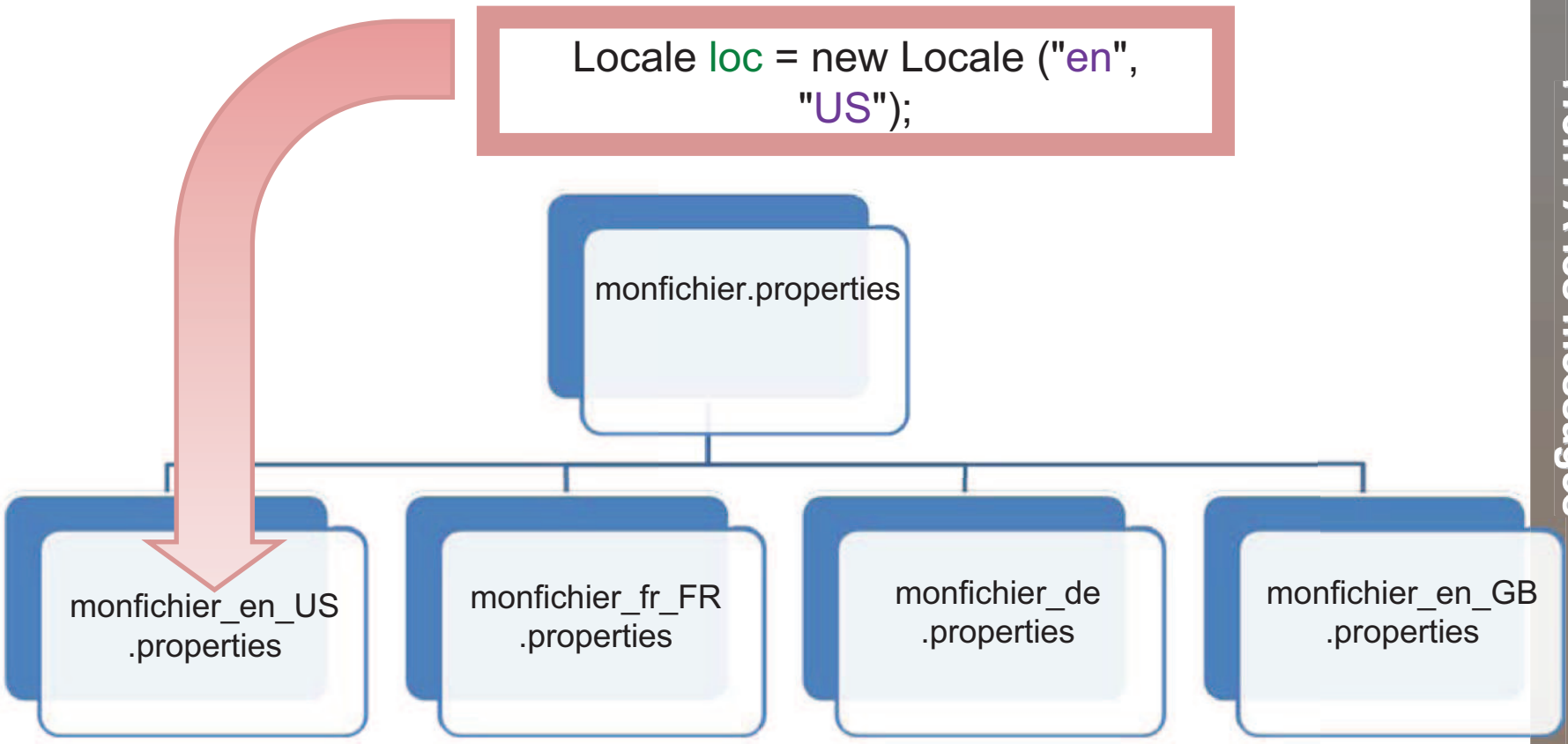
Étape 1 : les fichiers properties

➔ **Créer un fichier properties par langue/région**



Étape 2 : La Locale

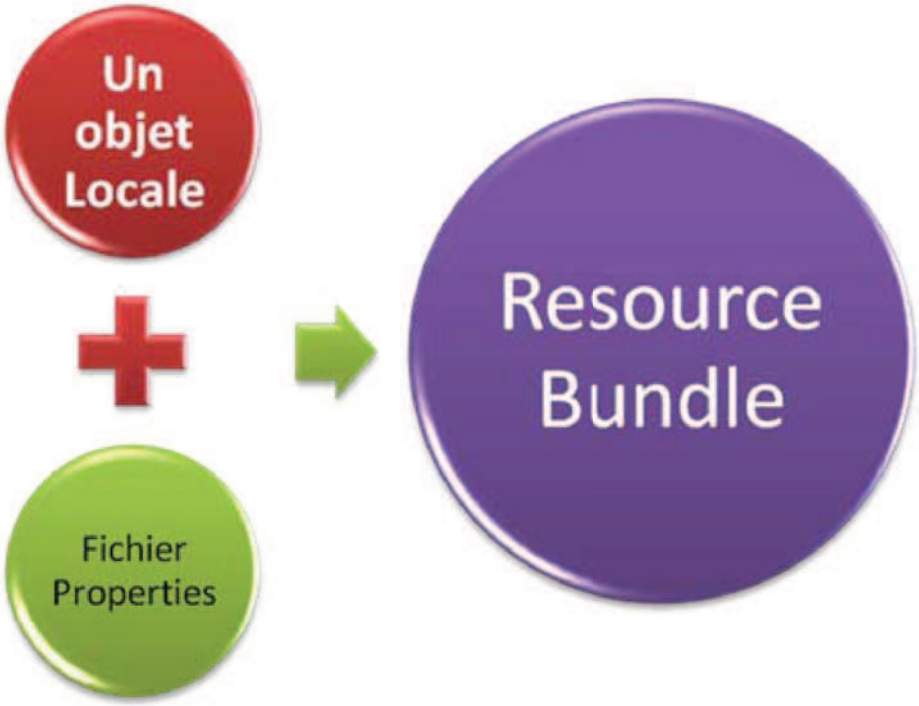
➔ **La locale permet de choisir la langue/region et donc le fichier resources que l'on va utiliser:**



Étape 3 : Resource Bundle

➔ **C'est par le ResourceBundle que l'on charge les données du fichier properties**

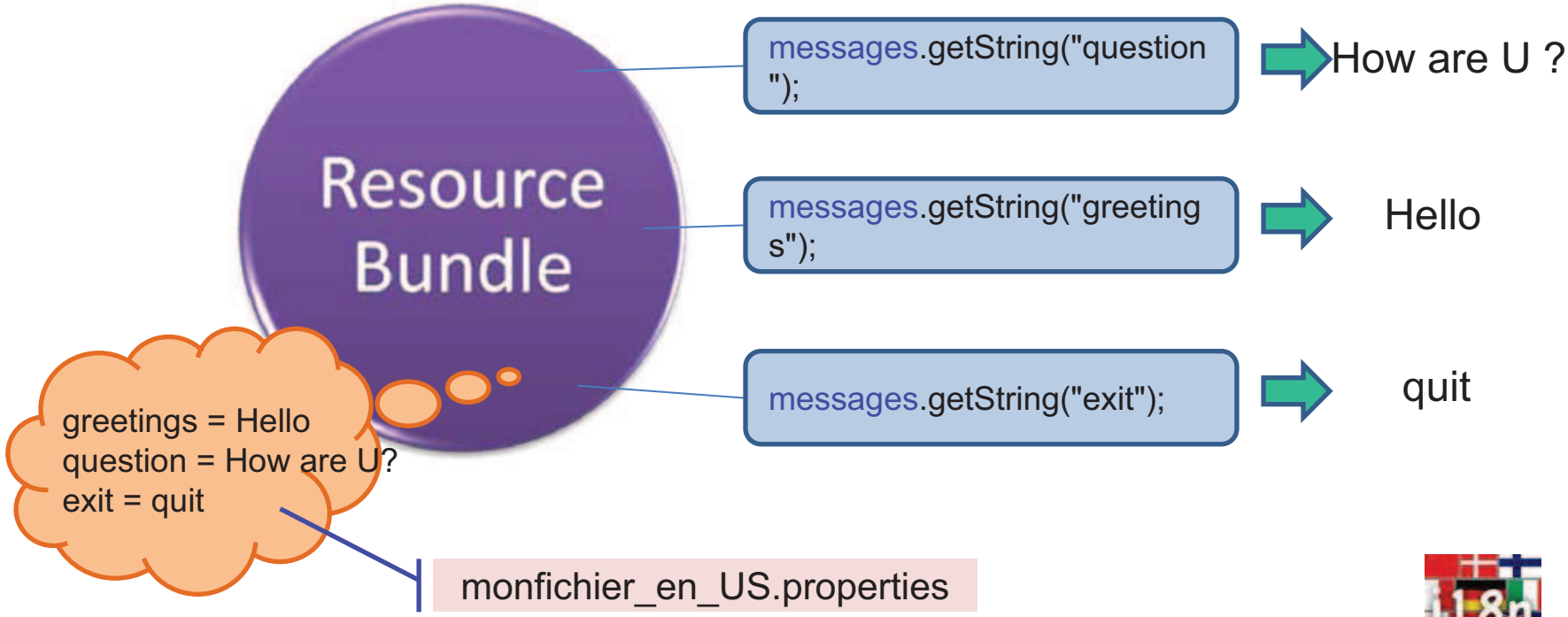
```
ResourceBundle messages;
messages = ResourceBundle.getBundle("monfichier", loc);
```



Étape 4 : Les messages

➔ **Reste ensuite à récupérer les messages**

```
String msg1 = messages.getString("greetings");
```



B : Le Formattage des dates, monnaies…

```
Locale locale = Locale.getDefault();

//Formatage d'une donnée de type numérique
NumberFormat number = NumberFormat.getNumberInstance( locale );
String str = number.format( new Double(12125.90) );

//Formatage d'une donnée monétaire
NumberFormat price = NumberFormat.getCurrencyInstance( locale );
String priceStr = price.format( new Double(1299.95) );

//Formatage d'une donnée de pourcentage
NumberFormat percent = NumberFormat.getPercentInstance( locale );
String percentStr = percent.format( new Double(50) );

//Formatage d'une donnée de type date avec (DateFormat)
DateFormat date = DateFormat.getDateInstance( DateFormat.DEFAULT, locale);
String dateStr = date.format(new Date());
```


i18n

- Pour récupérer la langue de la JVM ou changer par une autre:
  - Locale.getDefault();
  - Locale loc = new Locale("en", "US");
- Pour récupérer les valeurs de textes dans une langue
  - rs = ResourceBundle.getBundle("nompackage.nomfichier", loc);
  - String valeur = rs.getString("cle");
- Pour les dates:
  - la classe *DateFormat*
- Pour les monnaies, nombres et pourcentages :
  - la classe *NumberFormat*

RESUME