

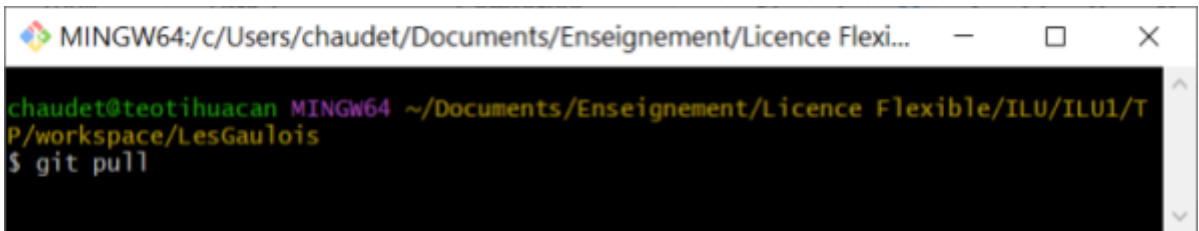
## Du code "propre"

### 1 Environnement GitHub (Rappel)

Mettre à jour votre dépôt local avec le contenu du dépôt distant.

Pour cela, il existe deux solutions : en ligne de commande ou avec l'outil graphique.

- En ligne de commande :
  - Avec un **explorateur de fichier** se placer dans le répertoire du projet 'LesGaulois' qui se trouve sous votre workspace (vous devez y trouver le fichier `.gitignore`), puis cliquer droit (sans rien sélectionner) et cliquer sur 'Git Bash Here'.
  - Dans Git Bash (ouvert directement dans le répertoire du projet 'LesGaulois'), tapez la commande `'git pull'`.



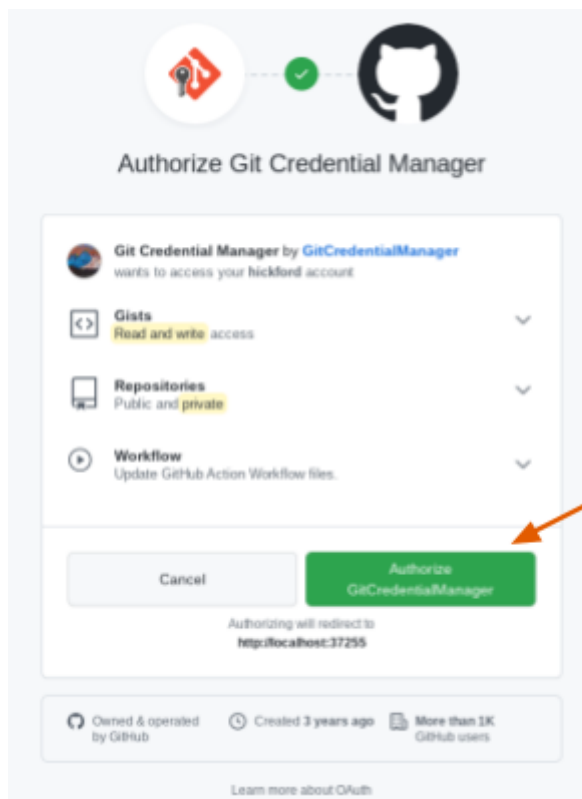
```
MINGW64:/c/Users/chaudet/Documents/Enseignement/Licence Flexi...  
chaudet@teotihuacan MINGW64 ~/Documents/Enseignement/Licence Flexible/ILU/ILU1/TP/workspace/LesGaulois  
$ git pull
```

- Avec l'outil graphique, le bouton 'Fetch origin' permet de mettre à jour la fenêtre principale pour connaître les modifications à mettre à jour localement. Il suffit ensuite de cliquer sur le bouton 'Pull origin'.

### Historisation périodique de votre projet.

- Avec un **explorateur de fichier** retrouver sous votre projet LesGaulois (vous devez y trouver le fichier `.gitignore`) puis cliquer droit (sans rien sélectionner) et sélectionner 'Git Bash Here'.
- Sous Git Bash : utiliser les commandes git :
  - `git add .`
  - `git commit -m <Intitulé des modifications>`  
**exemple :** `git commit -m "TP1 methode toString"`
  - `git push`

- Une fenêtre peut s'ouvrir (figure ci-contre) en vous demandant de vous identifier, Sélectionner "Sign in with your browser",
- Une autre fenêtre s'ouvre (extrait ci-dessous), sélectionner "Authorize GitCredentialManager"



**Authorize  
GitCredentialManager**

## 2 Mise en place de l'environnement

Vous allez insérer du code Java dans votre projet, malheureusement ce code à deux problèmes :

- Il est mal écrit : il ne respecte pas les bonnes pratiques de Java
- Il est buggé : à l'exécution nous n'obtenons pas le résultat attendu

Dans cette partie vous devez simplement insérer le code tel qu'on vous l'a donné sans penser à le modifier.

Dans la partie 3 nous nous intéresserons uniquement à respecter les bonnes pratiques en utilisant obligatoirement l'outil SonarLint.

Dans la partie 4 nous corrigerons les bugs en utilisant obligatoirement le debugger d'éclipse.

### *Modification de la classe "Romain"*

Ajouter l'attribut :

```
private String texte;
```

Mettre votre méthode recevoirCoup en commentaire : sélectionner toute la méthode et appuyer sur les touches : ctrl + shift + /

Copier/Coller la méthode suivante :

```
public Equipement[] recevoirCoup(int forceCoup) {
    Equipement[] equipementEjecte = null;
    // précondition
    assert force > 0;
    int oldForce = force;

    forceCoup = CalculResistanceEquipement(forceCoup);

    force -= forceCoup;
    // if (force > 0) {
    //     parler("Aïe");
    // } else {
    //     equipementEjecte = ejecterEquipement();
    //     parler("J'abandonne...");
    // }
    switch (force) {
    case 0:
        parler("Aïe");
```

```

    default:
        equipementEjecte = ejecterEquipement();
        parler("J'abandonne...");
        break;
    }
    // post condition la force a diminuée
    assert force < oldForce;
    return equipementEjecte;
}

```

Ajouter les deux méthodes suivantes :

```

private int CalculResistanceEquipement(int forceCoup) {
    texte = "Ma force est de " + this.force + ", et la force du
coup est de " + forceCoup;
    int resistanceEquipement = 0;
    if (!(nbEquipement == 0)) {
        texte += "\nMais heureusement, grace à mon équipement sa
force est diminué de ";
        for (int i = 0; i < nbEquipement;) {
            if ((equipements[i] != null &&
equipements[i].equals(Equipement.BOUCLIER)) == true) {
                resistanceEquipement += 8;
            } else {
                System.out.println("Equipement casque");
                resistanceEquipement += 5;
            }
            i++;
        }
        texte += resistanceEquipement + "!";
    }
    parler(texte);
    forceCoup -= resistanceEquipement;
    return forceCoup;
}

```

```

private Equipement[] ejecterEquipement() {
    Equipement[] equipementEjecte = new Equipement[nbEquipement];
    System.out.println("L'équipement de " + nom.toString() + "
s'envole sous la force du coup.");
    //TODO
    int nbEquipementEjecte = 0;
    for (int i = 0; i < nbEquipement; i++) {

```

```

        if (equipements[i] == null) {
            continue;
        } else {
            equipementEjecte[nbEquipementEjecte] =
equipements[i];
            nbEquipementEjecte++;
            equipements[i] = null;
        }
    }
    return equipementEjecte;
}

```

### Modification de la classe "Gaulois"

Ajouter les attributs suivants :

```

private int force, nb_trophees;
private Equipement trophées[] = new Equipement[100];

```

Mettre votre méthode *prendreParole* en commentaire : sélectionner toute la méthode et appuyer sur les touches : ctrl + shift + /

Copier/Coller la méthode suivante :

```

private String prendreParole() {
    String texte = "Le gaulois " + nom + " : ";
    return texte;
}

```

Mettre votre méthode *frapper* en commentaire : sélectionner toute la méthode et appuyer sur les touches : ctrl + shift + /

Copier/Coller la méthode suivante :

```

public void frapper(Romain romain) {
    System.out.println(nom + " envoie un grand coup dans la
mâchoire de " + romain.getNom());
    Equipement trophées[] = romain.recevoirCoup((force / 3) *
effetPotion);
    for (int i = 0; trophées != null && i < trophées.length; i++,
nb_trophees++) {
        this.trophées[nb_trophees] = trophées[i];
    }
    return;
}

```

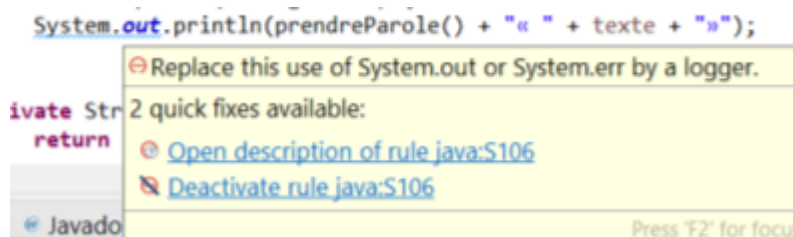
### 3 Coder proprement (utilisation de sonarLint)

#### Explications

Vous devrez corriger tout ce qui est souligné en bleu, excepté la règle sur le `System.out.println` :

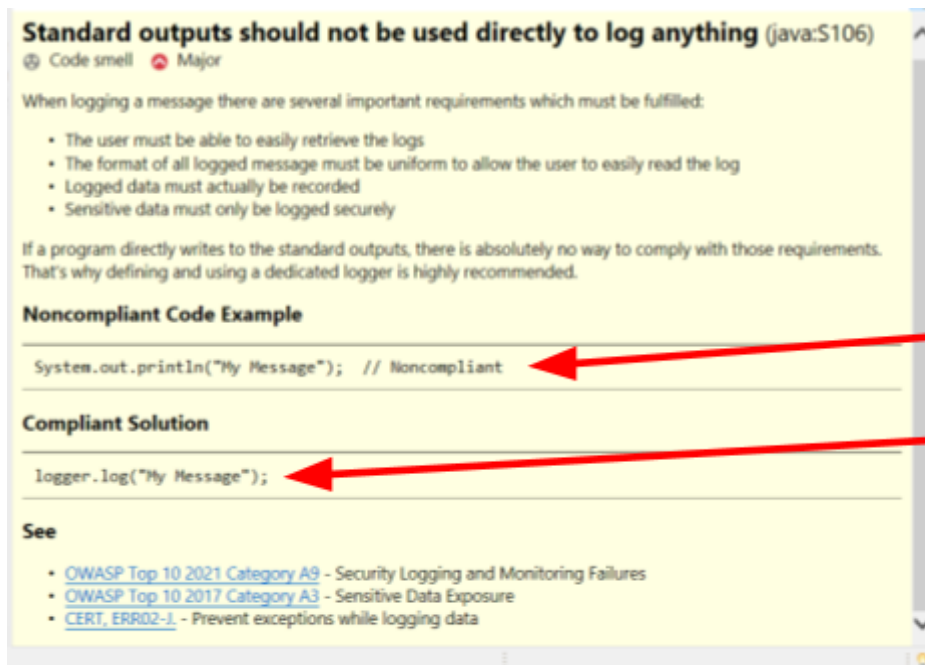
```
System.out.println(prendreParole() + "« " + texte + "»");
```

Pour visualiser la règle enfreinte, positionner le curseur sur la partie de code souligné en bleu.



puis cliquer sur `Open description rule java:S106`

Vous pouvez visualiser la règle :



**Standard outputs should not be used directly to log anything (java:S106)**

Code smell Major

When logging a message there are several important requirements which must be fulfilled:

- The user must be able to easily retrieve the logs
- The format of all logged message must be uniform to allow the user to easily read the log
- Logged data must actually be recorded
- Sensitive data must only be logged securely

If a program directly writes to the standard outputs, there is absolutely no way to comply with those requirements. That's why defining and using a dedicated logger is highly recommended.

**Noncompliant Code Example**

```
System.out.println("My Message"); // Noncompliant
```

**Compliant Solution**

```
logger.log("My Message");
```

**See**

- [OWASP Top 10 2021 Category A9](#) - Security Logging and Monitoring Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [CERT, ERR02-I](#) - Prevent exceptions while logging data

**Le genre d'erreur que vous avez fait**

**Le genre de solution que vous devez adopter**

Comme écrit précédemment, nous ne corrigerons pas cette erreur, mais vous devez corriger toutes les autres.

*Corrections à effectuer*

Les lignes données correspondent à mes copies d'écran, bien entendu vous devez faire la correspondance avec votre code.

## Classe Gaulois

```

5 public class Gaulois {
6     private String nom;
7     private int force, nb_trophées;
8     private int effetPotion = 1;
9     private Equipement trophées[] = new Equipement[100];

```

ligne 7 : S1659 et S116

ligne 9 : S1197

ATTENTION : Si vous souhaitez modifier le nom d'une classe / méthode / attribut ou variable local :

- sélectionner ce que vous souhaitez modifier
- cliquer droit dessus
- Refactor > Rename
- écrivez le nouveau nom puis tapez sur la touche entrer

```

24= private String prendreParole() {
25     String texte = "Le gaulois " + nom + " : ";
26     return texte;
27 }

```

ligne 25 : S1488

```

29= public void frapper(Romain romain) {
30     System.out.println(nom + " envoie un grand coup dans la mâchoire de " + romain.getNom());
31     Equipement trophées[] = romain.recevoirCoup((force / 3) * effetPotion);
32     for (int i = 0; trophées != null && i < trophées.length; i++, nb_trophées++) {
33         this.trophées[nb_trophées] = trophées[i];
34     }
35     return;
36 }

```

ligne 31 : S1197 et S1117

ligne 35 : S3626

```

56 // @Override
57 // public String toString() {
58 //     return "Gaulois [nom=" + nom + ", force=" + force + ", effetPotion=" + effetPotion + "]";
59 // }

```

ligne 57 : S125

## Classe Romain

```
3 public class Romain {
4
5     private String nom;
6     private int force;
7     private Equipement[] equipements = new Equipement[2];
8     private int nbEquipement = 0;
9     private String texte;
```

ligne 9 : S1450

```
34 public Equipement[] recevoirCoup(int forceCoup) {
35     Equipement[] equipementEjecte = null;
36     // précondition
37     assert force > 0;
38     int oldForce = force;
39
40     forceCoup = CalculResistanceEquipement(forceCoup);
41
42     force -= forceCoup;
43     // if (force > 0) {
44     //     parler("Aïe");
45     // } else {
46     //     equipementEjecte = ejecterEquipement();
47     //     parler("J'abandonne...");
48     // }
49     switch (force) {
50     case 0:
51         parler("Aïe");
52
53     default:
54         equipementEjecte = ejecterEquipement();
55         parler("J'abandonne...");
56         break;
57     }
58     // post condition la force à diminuer
59     assert force < oldForce;
60     return equipementEjecte;
61 }
```

Dans l'ordre :

ligne 50 : S128

ligne 49 : S1301



```
63 private int CalculResistanceEquipement(int forceCoup) {
64     texte = "Ma force est de " + this.force + ", et la force du coup est de " + forceCoup;
65     int resistanceEquipement = 0;
66     if (!(nbEquipement == 0)) {
67         texte += "\nMais heureusement, grace à mon équipement sa force est diminué de ";
68         for (int i = 0; i < nbEquipement; i++) {
69             if ((equipements[i] != null && equipements[i].equals(Equipement.BOUCLIER)) == true) {
70                 resistanceEquipement += 8;
71             } else {
72                 System.out.println("Equipement casque");
73                 resistanceEquipement += 5;
74             }
75             i++;
76         }
77         texte += resistanceEquipement + "!";
78     }
79     parler(texte);
80     forceCoup -= resistanceEquipement;
81     return forceCoup;
82 }
```

ligne 63 : S100

ligne 66 : S1940

ligne 69 : S1125

ligne 75 : S127

ligne 77 : S2757

```
109 private Equipement[] ejecterEquipement() {
110     System.out.println("L'équipement de " + nom.toString() + " s'envole sous la force du coup.");
111     Equipement[] equipementEjecte = new Equipement[nbEquipement];
112     //TODO
113     int nbEquipementEjecte = 0;
114     for (int i = 0; i < nbEquipement; i++) {
115         if (equipements[i] == null) {
116             continue;
117         } else {
118             equipementEjecte[nbEquipementEjecte] = equipements[i];
119             nbEquipementEjecte++;
120             equipements[i] = null;
121         }
122     }
123     return equipementEjecte;
124 }
```

ligne 111 : S1858

ligne 112 : S1135

ligne 116 : S3626

## 4 Utilisation du debugger

Télécharger depuis Moodle la classe "Scenario.java" et placer le dans le paquetage histoire.

Même en ayant corrigé toutes les erreurs de styles, l'application donne des résultats aberrants.

Si on lance plusieurs fois l'application on peut obtenir un résultat normal :

```
Le druide Panoramix : « Bonjour, je suis le druide Panoramix et ma potion peut aller d'une force 5 à 10.»
Le druide Panoramix : « Je vais aller préparer une petite potion...»
Le druide Panoramix : « J'ai préparé une super potion de force 8.»
Le druide Panoramix : « Non, Obélix !... Tu n'auras pas de potion magique !»
Le gaulois Obélix : « Par Bélénos, ce n'est pas juste !»
Le gaulois Astérix : « Merci Druide, je sens que ma force est 8 fois décuplée.»
Le gaulois Astérix : « Bonjour»
Le soldat Minus s'équipe avec un bouclier.
Le soldat Minus s'équipe avec un casque.
Le soldat Milexcus s'équipe avec un casque.
Le romain Minus : « UN GAU... UN GAUGAU...»
Astérix envoie un grand coup dans la mâchoire de Minus
Equiperment casque
Le romain Minus : « Ma force est de 6, et la force du coup est de 16
Mais heureusement, grace à mon équipement sa force est diminué de 13!»
Le romain Minus : « Aïe»
Astérix envoie un grand coup dans la mâchoire de Minus
Equiperment casque
Le romain Minus : « Ma force est de 3, et la force du coup est de 16
Mais heureusement, grace à mon équipement sa force est diminué de 13!»
L'équipement de Minus s'envole sous la force du coup.
Le romain Minus : « J'abandonne...»
Le romain Milexcus : « UN GAU... UN GAUGAU...»
Astérix envoie un grand coup dans la mâchoire de Milexcus
Equiperment casque
Le romain Milexcus : « Ma force est de 8, et la force du coup est de 16
Mais heureusement, grace à mon équipement sa force est diminué de 5!»
L'équipement de Milexcus s'envole sous la force du coup.
Le romain Milexcus : « J'abandonne...»
```

Ou un résultat aberrant (extrait) : plus le romain reçoit de coups, plus il a de force !

```
...
Astérix envoie un grand coup dans la mâchoire de Minus
Equiperment casque
Le romain Minus : « Ma force est de 6, et la force du coup est de 10
Mais heureusement, grace à mon équipement sa force est diminué de 13!»
Le romain Minus : « Aïe»
Astérix envoie un grand coup dans la mâchoire de Minus
Equiperment casque
Le romain Minus : « Ma force est de 9, et la force du coup est de 10
Mais heureusement, grace à mon équipement sa force est diminué de 13!»
...
```

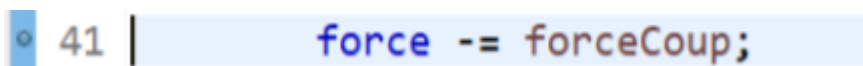
Pour corriger, nous allons utiliser le debugger d'Eclipse.

## Exemple d'utilisation

Vidéo d'exemple d'un débogage : <https://www.youtube.com/watch?v=oOOuCgUG6gE>  
ATTENTION cette vidéo est sonore, à éviter en salle de TP si vous ne possédez pas d'écouteur.

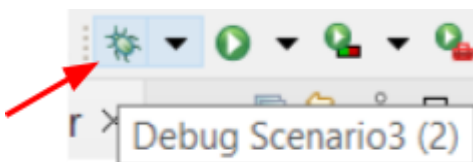
A priori il y a un problème sur la mise à jour de la force du romain.  
On va donc se positionner dans la classe "Romain" dans la méthode *recevoirCoup*  
et sur la ligne de l'instruction : `force -= forceCoup;`

Double-cliquer au niveau des numéros afin d'obtenir un point d'arrêt.

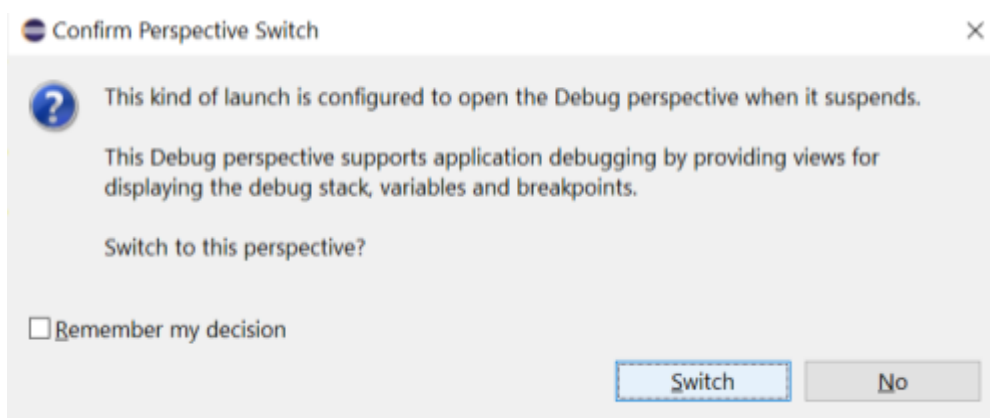


Un rond vert à gauche du numéro de la ligne apparaît.

Retourner sur votre scénario et au lieu de cliquer sur le bouton run, cliquer sur le bouton de débogage



Eclipse vous demande de changer de perspective : accepter.



**Ligne sur laquelle on se trouve dans chacune des classes en cours d'exécution**

**Consultation des valeurs des attributs et des variables de la méthode recevoirCoup**

**Le scénario est arrêté dès que le programme atteint le point d'arrêt**

Je peux connaître l'ensemble des valeurs des attributs et des variables en développant les arbres.

| Name                   | Value                 |
|------------------------|-----------------------|
| no method return value |                       |
| this                   | Romain (id=21)        |
| equipements            | Equipement[2] (id=23) |
| [0]                    | Equipement (id=32)    |
| name                   | "BOUCLIER" (id=35)    |
| nom                    | "bouclier" (id=36)    |
| ordinal                | 1                     |
| [1]                    | Equipement (id=34)    |
| name                   | "CASQUE" (id=39)      |
| nom                    | "casque" (id=40)      |
| ordinal                | 0                     |
| force                  | 6                     |
| nbEquipement           | 2                     |
| nom                    | "Minus" (id=25)       |
| forceCoup              | 5                     |
| equipementEjecte       | null                  |
| oldForce               | 6                     |

Je peux voir les valeurs des attributs du romain minus :

- Son tableau equipements possède 2 valeurs : [BOUCLIER,CASQUE]
- sa force est de 6
- il possède 2 équipements
- son nom est Minus

Je peux lire les variables locales de la méthode recevoirCoup :

- la force du coup reçu par Minus est de 5
- aucun de ses équipements n'ont été éjecté : null
- son ancienne force est de 6

Ensuite vous avez plusieurs possibilités :



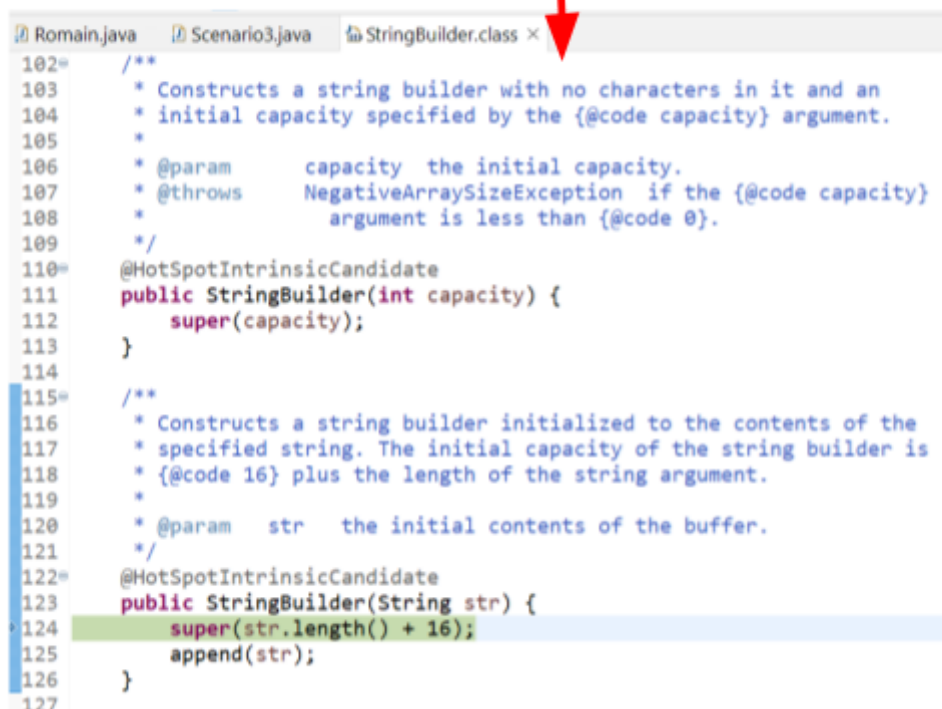
Dans l'ordre je peux :

- resume : aller au point d'arrêt suivant sans m'arrêter (je peux créer plusieurs points d'arrêts dans mon programme).
- stopper l'exécution du programme.
- continuer l'exécution pas à pas en entrant dans les différentes méthodes (parler, ejecterEquipement)
- Exécute la ligne courante et arrête l'exécution avant la ligne suivante sans entrer dans les différentes méthodes.
- Exécute le code de la ligne courante jusqu'à la prochaine instruction return de la méthode : dans la classe Gaulois

```
Equipement[]  trophéesBataille  =  romain.recevoirCoup((force  /  3)  *
effetPotion);
```

Si vous tombez sur des méthodes de la javadoc je vous conseille de passer au point suivant sans y entrer, car en l'absence de l'installation de la Javadoc vous passez dans des fichiers .class. Par exemple pour la méthode System.out.println

```
27=  public void frapper(Romain romain) {
28  System.out.println(nom + " envoie un grand coup dans la mâchoire de " + romain.getNom());
```





## A vous de trouver le bug


En jouant le scénario plusieurs fois, vous allez vous apercevoir que c'est bien la méthode *calculResistanceEquipement* qui pose problème.

Exemple d'un extrait de l'affichage de la console

```
Astérix envoie un grand coup dans la mâchoire de Minus
Equipement casque
Le romain Minus : « Ma force est de 6, et la force du coup est de 10
Mais heureusement, grace à mon équipement sa force est diminué de 13! »
Le romain Minus : « Aïe »
Astérix envoie un grand coup dans la mâchoire de Minus
Equipement casque
Le romain Minus : « Ma force est de 9, et la force du coup est de 10
Mais heureusement, grace à mon équipement sa force est diminué de 13! »
```

La force du romain minus augmente de 6 à 9 !

Placer des points d'arrêt bien choisis dans la méthode *calculResistanceEquipement* et trouver le bug !

Une fois le bug trouvé et corrigé, revenir à la perspective JAVA en appuyant sur le bouton  en haut à gauche.

### Bug supplémentaire

Dans la méthode *recevoirCoup* l'assertion "assert force < oldForce;" n'est plus systématiquement vérifiée. Etapes :

- trouver pourquoi,
- faire parler le romain en conséquence dans la méthode *recevoirCoup*,
- ajouter un boolean *vainqueur* dans la classe **Romain**, celui-ci est modifié dans la méthode *recevoirCoup*,
- modifier la condition d'arrêt de la boucle dans le scénario.

## 5 Préparation vers une utilisation de OCaml

### 1. Bienvenue au Musee

Les gaulois possèdent un musée dans lequel ils peuvent déposer les équipements des gaulois gagnés au combat. Ces équipements deviennent pour eux des trophées.

#### a) La classe "Trophee"

Créer la classe "Trophee" contenant deux attributs :

- gaulois de type Gaulois,
- équipement de type Equipement.

Générer le constructeur initialisant les 2 attributs

Générer les 2 getteurs

Ecrire la méthode `donnerNom` qui retourne le nom du gaulois.

#### b) La classe "Musee"

Créer la classe "Musee" contenant deux attributs :

- un tableau *trophées* pouvant contenir 200 trophées,
- un entier *nbTrophee* pour compter le nombre de trophées contenus dans le musée.

Créer la méthode *donnerTrophées* :

- prend en paramètre d'entrée le gaulois qui fait un don au musée et l'équipement qu'il donne,
- place le nouveau trophée dans le tableau *trophées*.

#### c) La classe Gaulois

Créer la méthode *faireUneDonnation* :

- prends en paramètre d'entrée le musée à qui le gaulois veut faire une donation,
- Si le gaulois possède des trophées dans son tableau *trophées* alors il les donne tous au musée en faisant une annonce comme par exemple :

```
Le gaulois Astérix : « Je donne au musee tous mes trophées :  
- bouclier  
- casque  
- casque»
```

## 2. Vers une analyse des données

Nous souhaitons que toutes les données stockées dans le tableau de la classe "Musee" puissent être analysées en OCaml.

Définir la méthode "extraireInstructionsOCaml" qui retourne sous la forme d'une chaîne de caractères la déclaration OCaml de la variable OCaml "musee" contenant la liste des couples de chaîne de caractères (nom du gaulois, nom du trophée) pris dans la variable Java "trophees".

On retournera par exemple la chaîne de caractères contenant le texte suivant :

```
let musee = [  
    "Astérix", "bouclier";  
    "Astérix", "casque";  
    "Astérix", "casque"  
]
```

Dans un TP de OCaml vous devrez traiter cette chaîne afin d'obtenir le nombre de casques que le musée possède pour chacun des gaulois.