

Nell TELECHEA-LOJOU

BUT2 Informatique Fontainebleau

Benjamin BRIBANT

RAPPORT BAKE

SAE DEV 3.2

TABLE DES MATIERES

Contenu

- I. Introduction
- II. Description des fonctionnalités
- III. Structure du programme
- IV. Graphe des dépendances
- V. Algorithme de détection des dépendances circulaires
- VI. Structures de données abstraites
- VII. Conclusion

I. Introduction

L'utilitaire de compilation BAKE est une version basique de la commande make qui permet de générer des fichiers et de les maintenir à jour par rapport à leurs sources. Bake est configuré grâce à un fichier nommé Bakefile qui contient des règles de dépendances entre les fichiers et leurs recettes de génération. Il peut également contenir des commentaires et des affectations de variables.

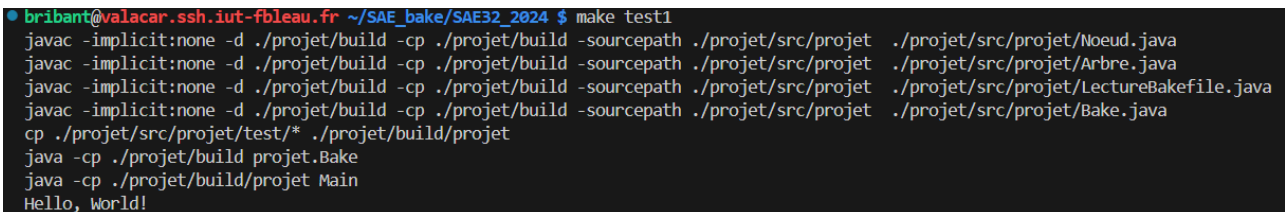
DESCRIPTION DES FONCTIONNALITES

II. Description des fonctionnalités

Le programme peut ou non prendre des fichiers en argument. Il peut aussi prendre un “-d” en option qui permet d’afficher des informations de débogage. Si un fichier source a été modifié après une première compilation, le programme ne recompile que celui-ci. Si le programme détecte une dépendance circulaire, il ignore la recette concernée et ne l’exécute pas.

Si l’exécution d’une recette échoue, le programme s’arrête.

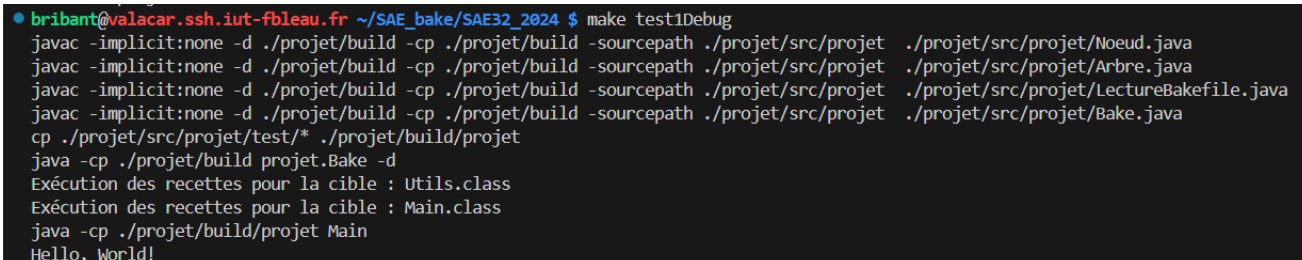
Utilisation de Bake sans argument (1ère compilation) :



```
• bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make test1
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp ./projet/src/projet/test/* ./projet/build/projet
java -cp ./projet/build projet.Bake
java -cp ./projet/build/projet Main
Hello, World!
```

Le programme compile puis exécute le résultat : Hello, World!

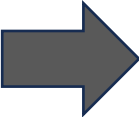
Utilisation de Bake avec option de débogage :



```
• bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make test1Debug
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp ./projet/src/projet/test/* ./projet/build/projet
java -cp ./projet/build projet.Bake -d
Exécution des recettes pour la cible : Utils.class
Exécution des recettes pour la cible : Main.class
java -cp ./projet/build/projet Main
Hello, World!
```

Comme c’est une première execution sans fichiers sources fournis, il n’y a pas de comparaison, mais le débogage affiche quelles recettes de quelle cible sont exécutées. Le résultat est le même qu’avant : Hello, World!

DESCRIPTION DES FONCTIONNALITES



```
● bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make testExistantDebug
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp -p ./projet/src/projet/testExistant/* ./projet/build/projet
java -cp ./projet/build projet.Bake -d
Le fichier ./projet/build/projet/Utils.class a été modifié pour la dernière fois le : 16/01/2025 17:17:02
Le fichier ./projet/build/projet/Utils.java a été modifié pour la dernière fois le : 16/01/2025 17:15:37
La cible Utils.class est déjà à jour.
Le fichier ./projet/build/projet/Main.class a été modifié pour la dernière fois le : 16/01/2025 17:17:02
Le fichier ./projet/build/projet/Main.java a été modifié pour la dernière fois le : 16/01/2025 17:14:51
La cible Main.class est déjà à jour.
java -cp ./projet/build/projet Main
Hello, World!
```

Les fichiers .class étant déjà là (2ème compilation), on les compare au .java pour voir s'ils sont plus récents, le processus est affiché avec le débogage.

Utilisation de Bake sans argument (2ème compilation, 1 fichier source modifié) :

```
● bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make testModifDebug
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp -p ./projet/src/projet/testModif/* ./projet/build/projet
java -cp ./projet/build projet.Bake -d
Le fichier ./projet/build/projet/Utils.class a été modifié pour la dernière fois le : 16/01/2025 18:20:37
Le fichier ./projet/build/projet/Utils.java a été modifié pour la dernière fois le : 16/01/2025 18:20:04
La cible Utils.class est déjà à jour.
Le fichier ./projet/build/projet/Main.class a été modifié pour la dernière fois le : 16/01/2025 18:20:31
Le fichier ./projet/build/projet/Main.java a été modifié pour la dernière fois le : 16/01/2025 18:20:54
Le fichier source .java (./projet/build/projet/Main.java) est plus récent que la cible.
Exécution des recettes pour la cible : Main.class
java -cp ./projet/build/projet Main
Hello, World!
```

Ici, on compare les fichiers sources, pour le Main, le .java est plus récent que le .class donc on recompile

DESCRIPTION DES FONCTIONNALITES

Utilisation de Bake avec un fichier en argument :

```
● bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make testArgumentDebug
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp -p ./projet/src/projet/testArgument/* ./projet/build/projet
java -cp ./projet/build projet.Bake -d Utils.class
Exécution des recettes pour la cible : Utils.class
```

Ici, on lui donne Utils.class en argument, il n'exécute que les recettes de Utils.class.

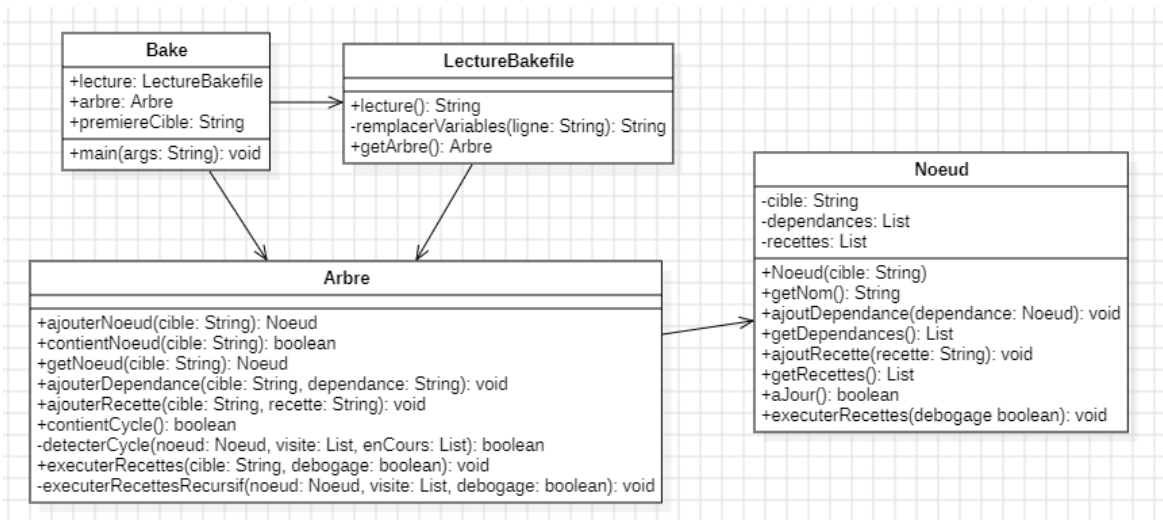
Utilisation de Bake avec une dépendance circulaire :

```
● bribant@valacar.ssh.iut-fbleau.fr ~/SAE_bake/SAE32_2024 $ make testCycle
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Noeud.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Arbre.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/LectureBakefile.java
javac -implicit:none -d ./projet/build -cp ./projet/build -sourcepath ./projet/src/projet ./projet/src/projet/Bake.java
cp ./projet/src/projet/testCycle/* ./projet/build/projet
java -cp ./projet/build projet.Bake
Cibles ignorées en raison de cycles :
- Utils.class
- Utils.class
- ElCycle.class
java -cp ./projet/build/projet Main
Hello, World!
Soy el méchant cycle >:D ! Hasta la vista baby !
¡Ay, caramba!
```

STRUCTURE DU PROGRAMME

III. Structure du programme

Diagramme de classe du projet



Notre programme est constitué de 4 classes. La classe `Bake` est la classe principale, elle contient le `main`. Elle récupère les arguments de la commande.

La classe `LectureBakefile` va lire le fichier `Bakefile` et stocker ses données (cibles, dépendances et recettes) dans un arbre représenté dans les classes `Arbre` et `Noeud`.

La classe `Arbre` s'occupe de la gestion de l'arbre, d'ajouter des nœuds, des recettes, de détecter de dépendances circulaires et d'exécuter des recettes. La classe `Noeud` s'occupe de la gestion d'un nœud de l'arbre, il est représenté par une cible qui a ses dépendances et ses recettes. La classe permet de gérer l'ajout d'un nœud, de ses dépendances et ses recettes. On peut aussi vérifier si la cible est à jour et exécuter ses recettes.

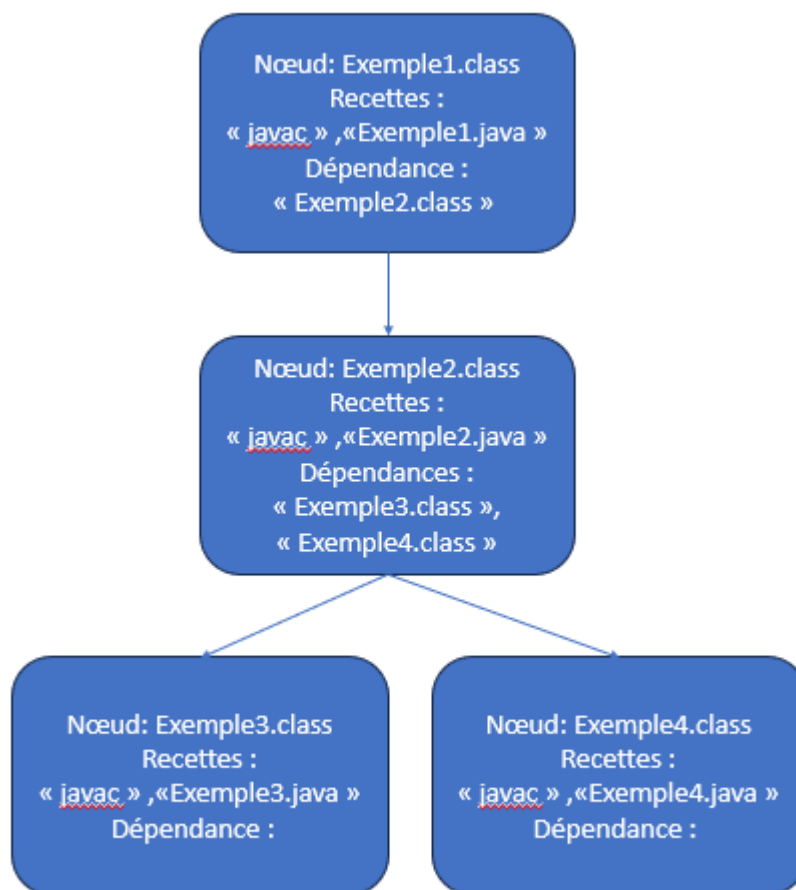
STRUCTURE DU PROGRAMME

L'exécution des recettes est en deux parties, elle est lancée dans l'arbre pour être appliquée sur chaque nœud.

IV. Graphe des dépendances

Les classes participant au graphe des dépendances sont les classes Arbre et Noeud. Lors de la lecture du Bakefile, un arbre est créé puis pour chaque règle nous créons un noeud dans lequel sont stockées les recettes de la règle ainsi que ses dépendances à d'autres noeuds.

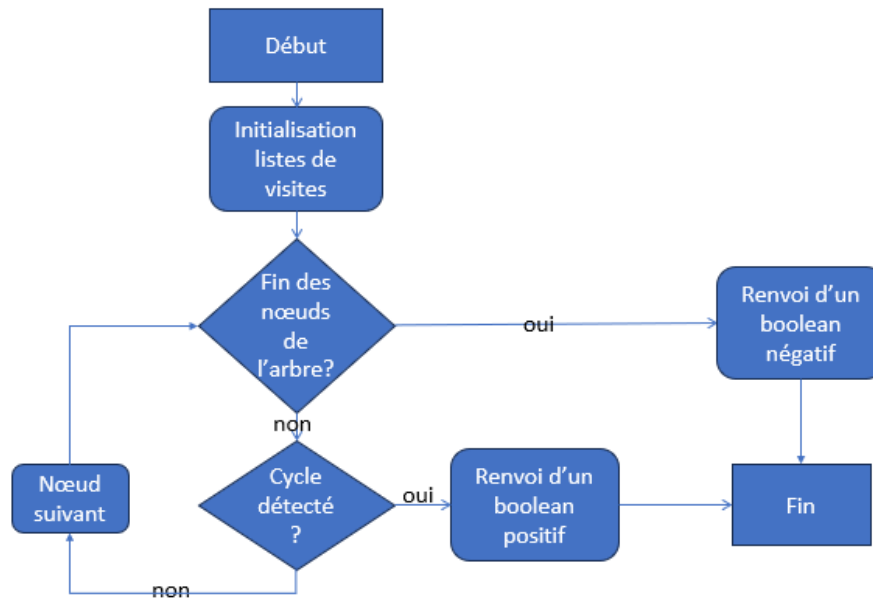
Voici un exemple d'arbre de la classe Arbre, il contient 4 objets de la classe Noeud :



DETECTION DES DEPENDANCES CIRCULAIRES

V. Algorithme de détection des dépendances circulaires

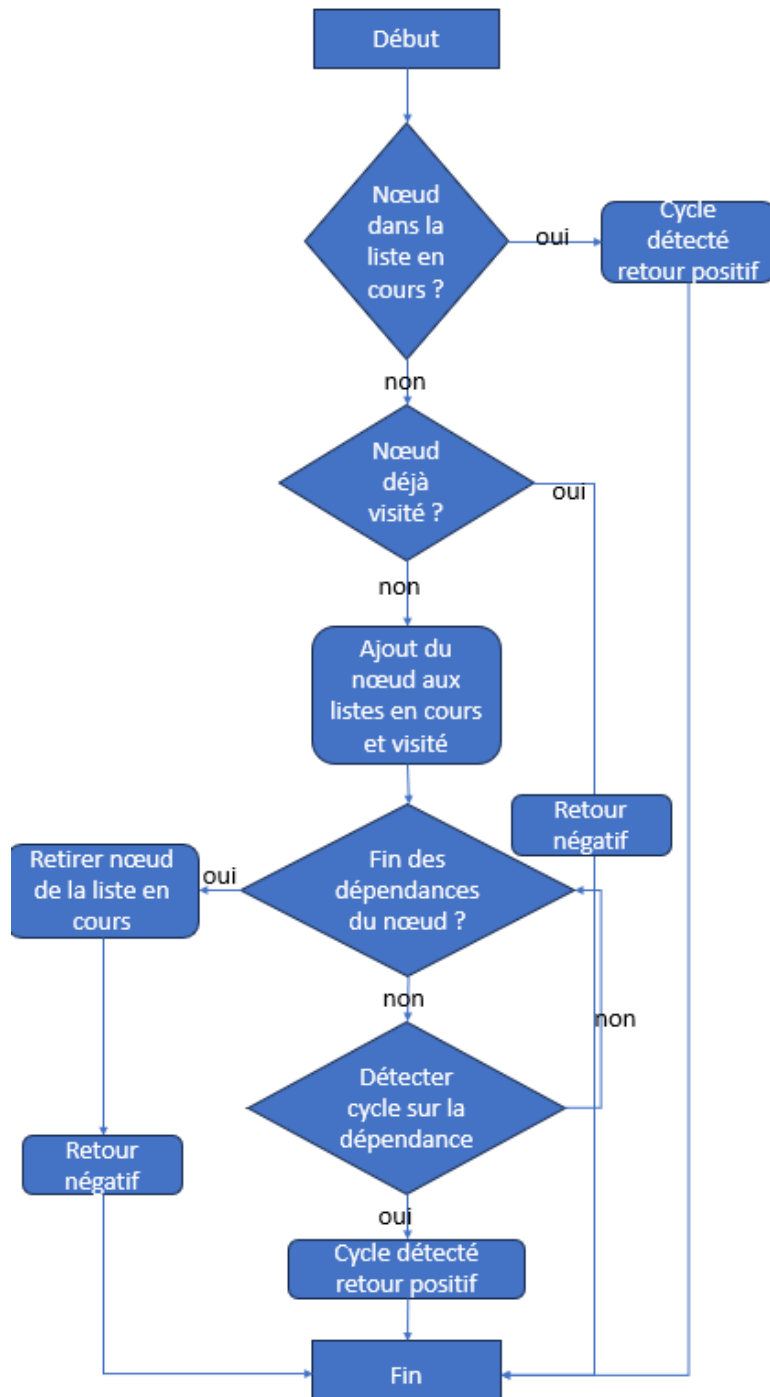
Diagramme d'activité de détection de dépendance circulaire



Tout d'abord, on initialise une liste qui va permettre de contenir les noeuds déjà visités lors de la visite en cours et une liste qui va contenir les noeuds déjà visités entièrement. Pour chaque noeud de l'arbre, on applique la méthode qui permet de détecter un cycle, s'il n'y a rien on continue de boucler sur les noeuds, sinon c'est qu'un cycle est détecté.

DETECTION DES DEPENDANCES CIRCULAIRES

Détection du cycle



DETECTION DES DEPENDANCES CIRCULAIRES

Dans la méthode de détection, on récupère les listes évoquées précédemment. On va vérifier si le nœud n'a pas déjà été visité dans la visite en cours, si c'est le cas c'est qu'il y a un cycle. Sinon on va ensuite vérifier que le nœud n'a pas déjà été vérifié dans une autre visite, si c'est le cas, on sort de la méthode, sinon on ajoute le nœud dans les deux listes pour indiquer qu'il a déjà été visité. Ensuite on appelle récursivement cette méthode sur toutes les dépendances du nœud. Si rien n'a été détecté, c'est qu'il n'y a pas de dépendance circulaire.

VI. Structures de données abstraites

Dans notre projet nous avons utilisé plusieurs structures de données abstraites. La principale étant notre arbre de dépendances présenté plus tôt dans ce rapport.

Lors de la lecture du Bakefile, nous stockons les variables du fichier dans un dictionnaire pour les substituer lors de la lecture des règles.

Les nœuds de l'arbre sont stockés dans un dictionnaire avec comme clé le nom du nœud et chaque nœud possède deux listes qui contiennent respectivement les dépendances et les recettes du nœud.

Dans le cas de dépendances circulaires, nous rangeons les recettes des règles qui posent problème dans une liste de règles à ignorer lors de la compilation des recettes. Et comme évoqué lors de l'explication de l'algorithme de détection, nous utilisons deux listes.

VII. Conclusion

Nell Telechea :

Concevoir cet outil de Make (Bake) a été intéressant et a permis de mieux comprendre le fonctionnement de Make et de pouvoir évidemment appliquer ce qu'on a vu en cours, comme à chaque saé. Contrairement à la dernière saé de Dorfromantik, nous avons essayé de mieux nous organiser, ce qui a permis de tout finir correctement à temps.

Benjamin Bribant :

Pour ma part, j'ai aimé faire ce projet qui était complètement différent des précédents. Cela m'a permis de voir plus en détail comment fonctionne réellement la commande make, et c'était très intéressant d'en faire notre propre version, même si moins complète. Cette fois-ci, nous nous y sommes pris à l'avance et avons eu le temps de vraiment finir le projet dans les temps. C'était un mode de travail beaucoup moins stressant que de devoir tout faire et corriger tous les problèmes à la dernière minute.