**INTRODUCTION**

Notre travail consistait dans un premier temps à créer un compte sur [www.github.com](http://www.github.com), d’y créer un dépôt, d’inviter des collaborateurs (d’autres membres du groupe), ensuite de chercher un projet dans un langage au choix, de discuter dessus entre collaborateurs, et de faire les dépôts de nos contributions dans le repository .Nous allons par la suite vous présenter la procédure que nous avons utilisée pour faire notre travail, le résultat des métriques appliquées à notre logiciel et la conclusion.

1. **PROCEDURE**

Pour atteindre nos objectifs, nous avons créé des comptes sur github où le compte maître est intitulé **carineakame,**  nous avons créé notre repository sous le nom **groupe 6**, nous avons téléchargé et installé la version desktop de github et nous l’avons cloné avec notre repository hébergé dans le site github, ensuite nous avons choisi un projet de gestion d’un supermarché conçu en langage Java, nous avons utilisé les plugings de Netbeans IDE pour analyser notre projet.

1. **METRIQUES**

**Définition des sigles: LOC= Line Of Code LCOM= Lack of COhesion in Methods**

**LOC**

Total LOC: 564 Classes LOC: Gestion: 564

Packages LOC: src: 564 private: 0 javaapplication3: 0

classes: 0 nbproject: 0

------------------

**LINES WITH IMPORTS**

------------------

Total imports: 4 Classes imports: Gestion: 4 Packages imports:

src: 4 private: 0 javaapplication3: 0

classes: 0 nbproject: 0

**------------------**

**BLANK LINES**

**------------------**

Total blank lines: 75 Classes blank lines: Gestion: 75 Packages blank lines:

src: 75 private: 0 javaapplication3: 0

classes: 0 nbproject: 0

**------------------**

**CLASSES COUNT**

**------------------**

Total classes: 1 Packages with the biggest number of classes:

src: 1 private: 0 javaapplication3: 0

classes: 0 nbproject: 0

**------------------**

**METHODS COUNT**

**------------------**

Total methods: 5 Classes with the biggest number of methods: Gestion: 5

**------------------**

**CYCLOMATIC COMPLEXITY**

**------------------**

Average cyclomatic complexity: 11.4 Methods with the highest cyclomatic complexity:

Gestion::main: 45

Gestion::getInt: 4

Gestion::getRayon: 4

Gestion::getLine: 2

Gestion::getDouble: 2

**------------------**

**LCOM**

**------------------**

**AVERAGE LCOM 1: 0**

Classes with the highest LCOM 1: Gestion: 0

Packages with the highest average LCOM 1: src: 0

private: 0 javaapplication3: 0 classes: 0 nbproject: 0

**------------------**

**AVERAGE LCOM 2: 0.0**

**------------------**

Classes with the highest LCOM 2: Gestion: 0.0

Packages with the highest average LCOM 2: src: 0.0

private: 0.0 javaapplication3: 0.0 classes: 0.0 nbproject: 0.0

**------------------**

**AVERAGE LCOM 3: 0.0**

**------------------**

Classes with the highest LCOM 3: Gestion: 0.0

Packages with the highest average LCOM 3: src: 0.0 private: 0.0

javaapplication3: 0.0 classes: 0.0 nbproject: 0.0

**------------------**

**AVERAGE LCOM 4: 1**

**------------------**

Classes with the highest LCOM 4: Gestion: 1 Packages with the highest average LCOM 4:

src: 1 private: 0 javaapplication3: 0 classes: 0 nbproject: 0

**CONCLUSION**

Au terme de notre travail, nous pouvons dire que notre logiciel de gestion d’un supermarché est exact car il fonctionne selon les spécifications, son code ne sent pas car ne contient pas les erreurs, il n’est pas assez maintenable ni intègre mais il est convivial