

Kelas : 02

Nomor Kelompok : 08

Nama Kelompok : AFAKA

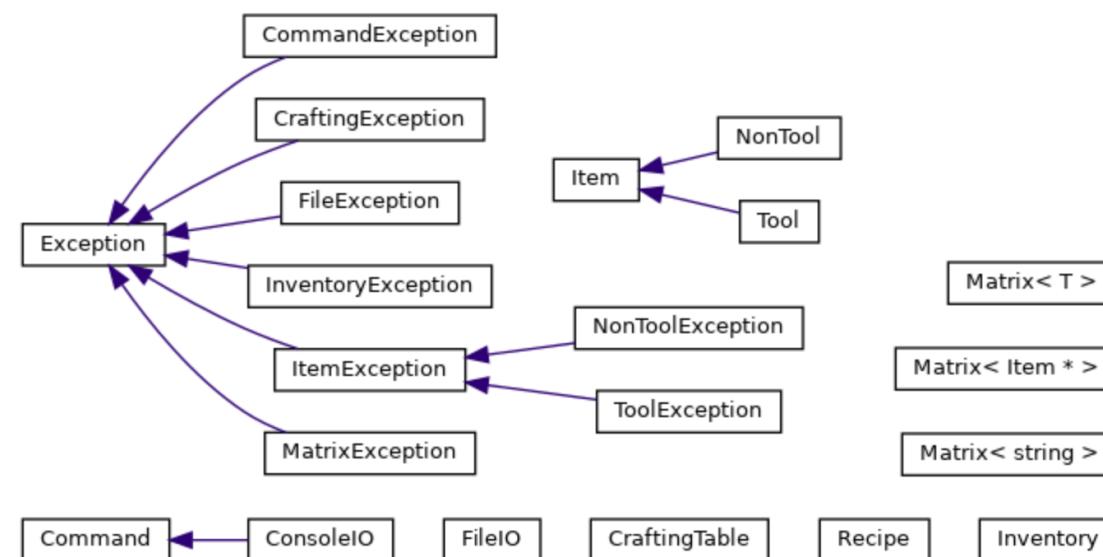
1. 13520023 / Ahmad Alfani Handoyo
2. 13520044 / Adiyansa Prasetya Wicaksana
3. 13520056 / Fikri Khoiron Fadhila
4. 13520065 / Rayhan Kinan Muhamnad
5. 13520083 / Sarah Azka Arief

Asisten Pembimbing : Gregorius Jovan Kresnadi

Link Repository : <https://github.com/rayhankinan/Tubes01-OOP>

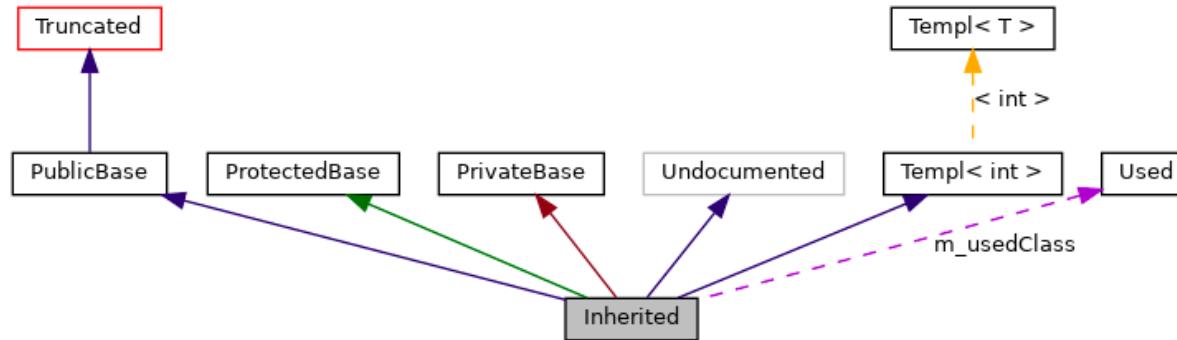
1. Diagram Kelas

a. Class Hierarchy



b. Class Diagram

Graph Legend

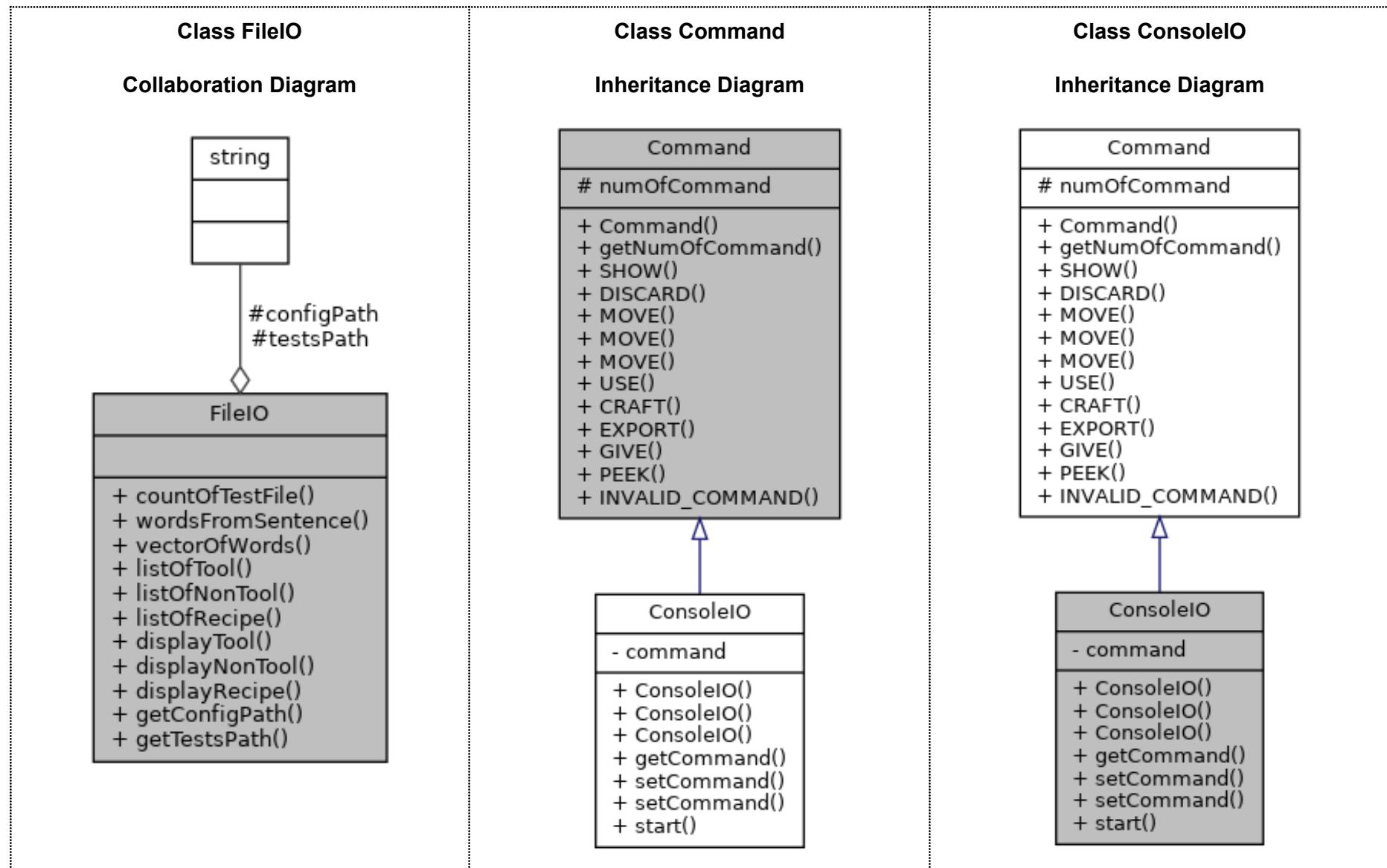


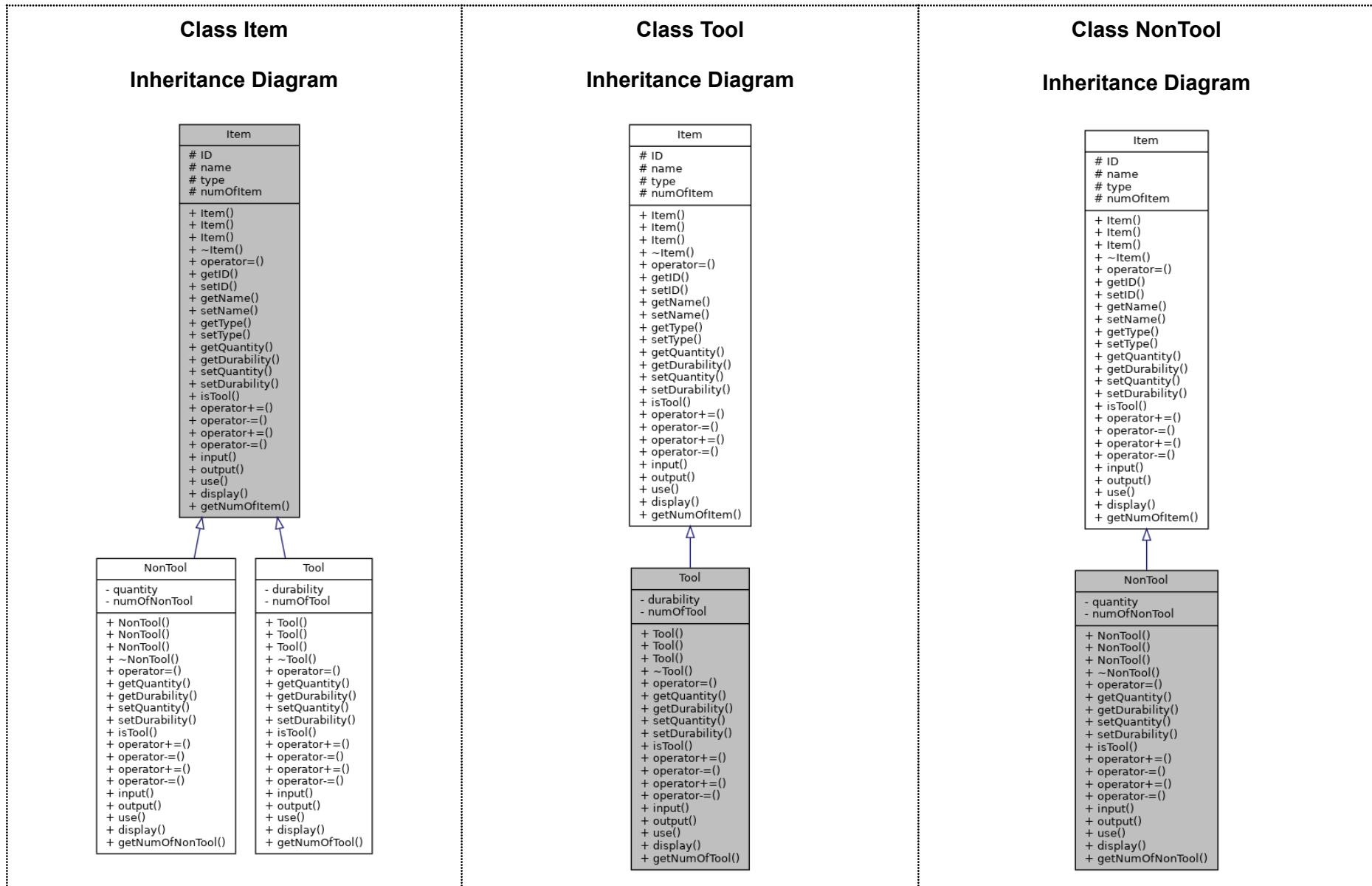
The boxes in the above graph have the following meaning:

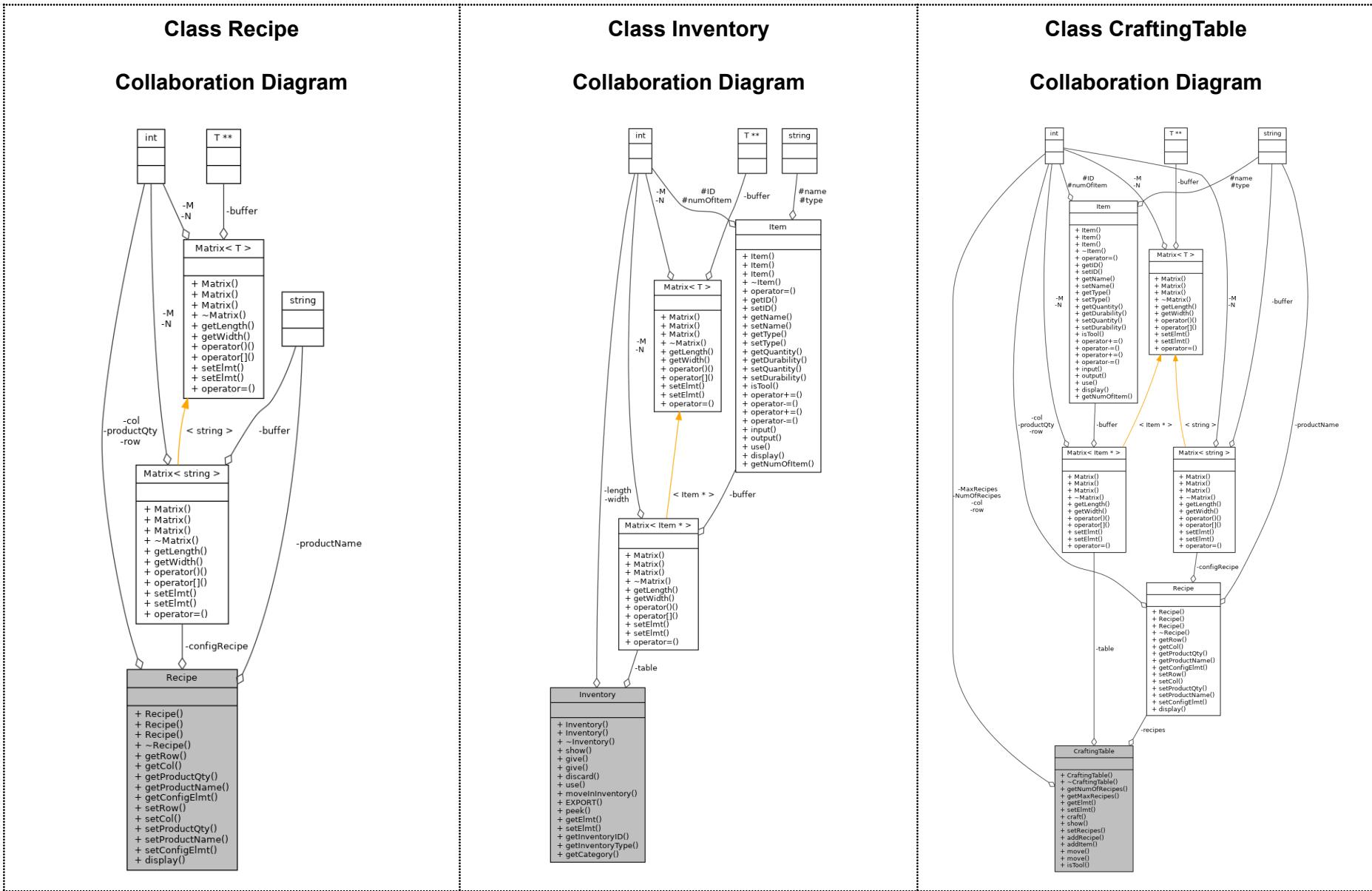
- A filled gray box represents the struct or class for which the graph is generated.
- A box with a black border denotes a documented struct or class.
- A box with a gray border denotes an undocumented struct or class.
- A box with a red border denotes a documented struct or class for which not all inheritance/containment relations are shown. A graph is truncated if it does not fit within the specified boundaries.

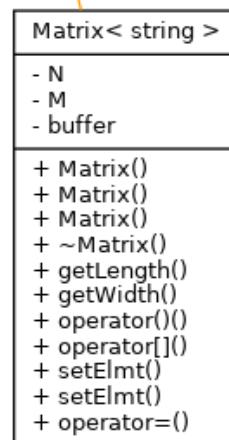
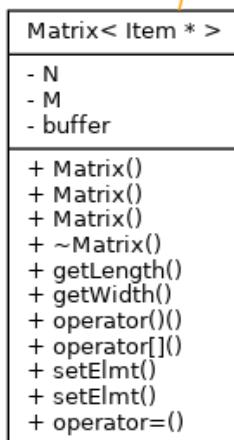
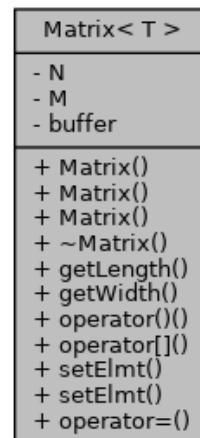
The arrows have the following meaning:

- A dark blue arrow is used to visualize a public inheritance relation between two classes.
- A dark green arrow is used for protected inheritance.
- A dark red arrow is used for private inheritance.
- A purple dashed arrow is used if a class is contained or used by another class. The arrow is labelled with the variable(s) through which the pointed class or struct is accessible.
- A yellow dashed arrow denotes a relation between a template instance and the template class it was instantiated from. The arrow is labelled with the template parameters of the instance.

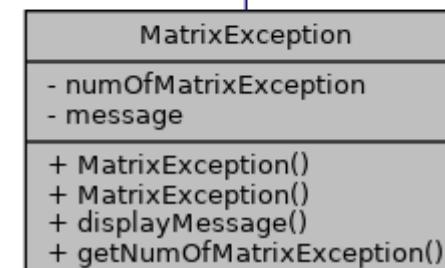
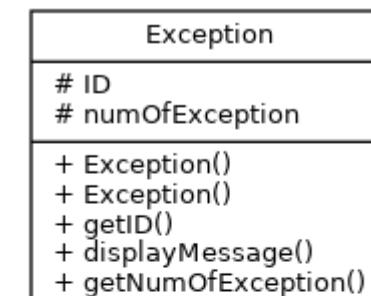
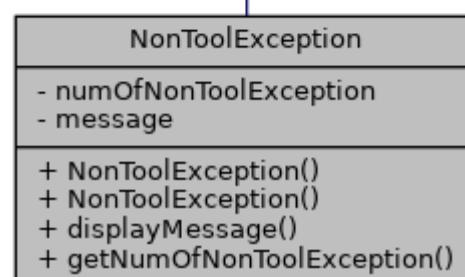
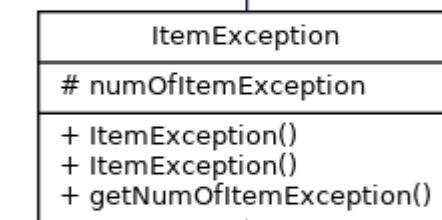
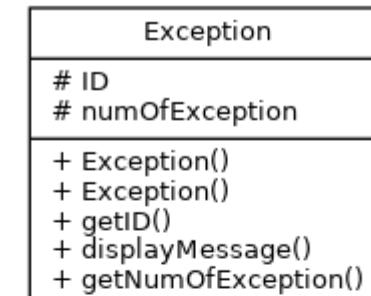


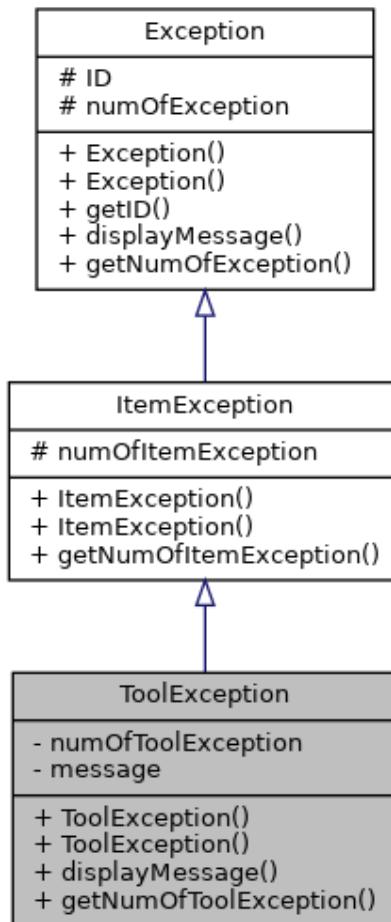
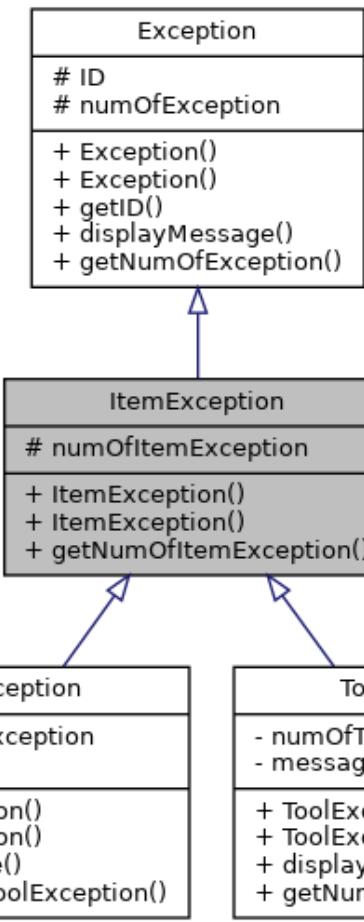
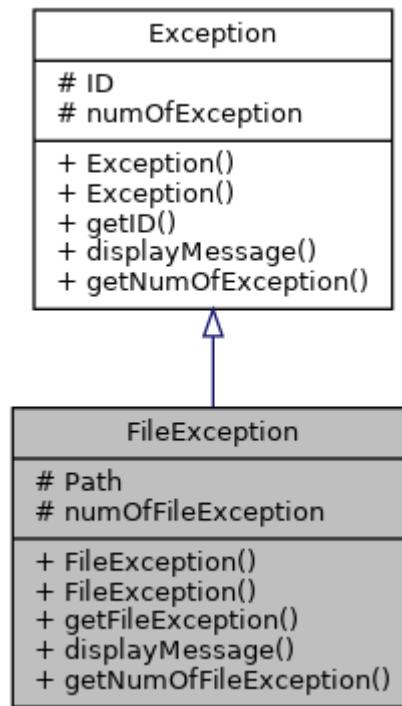


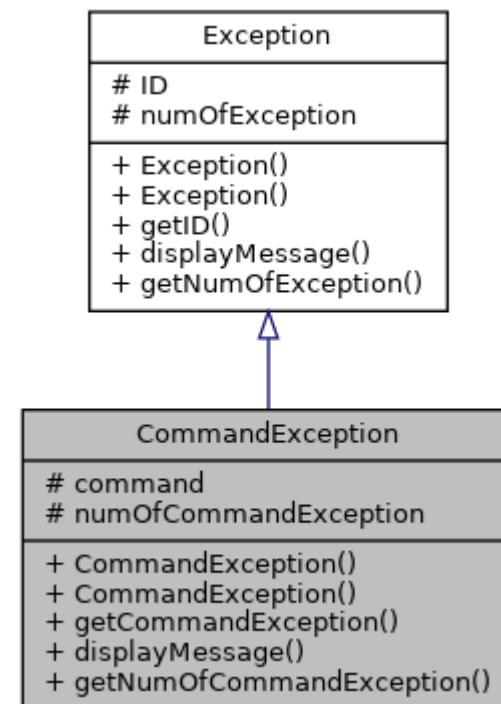
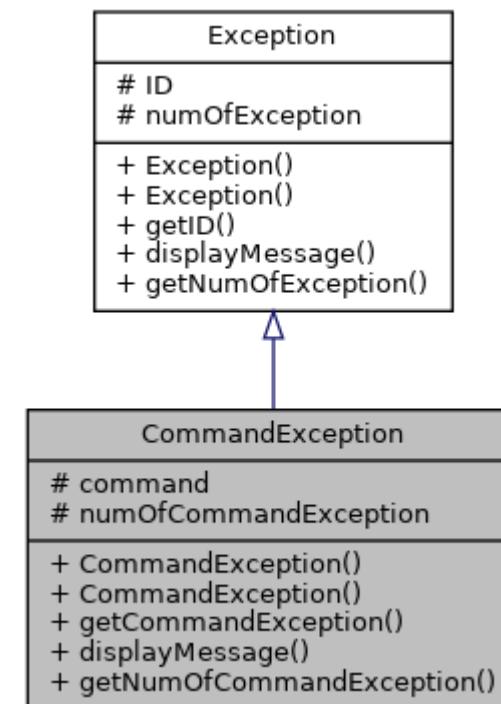
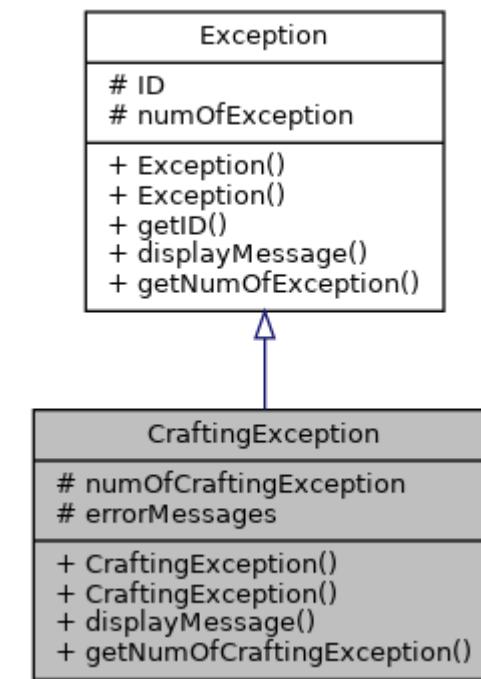


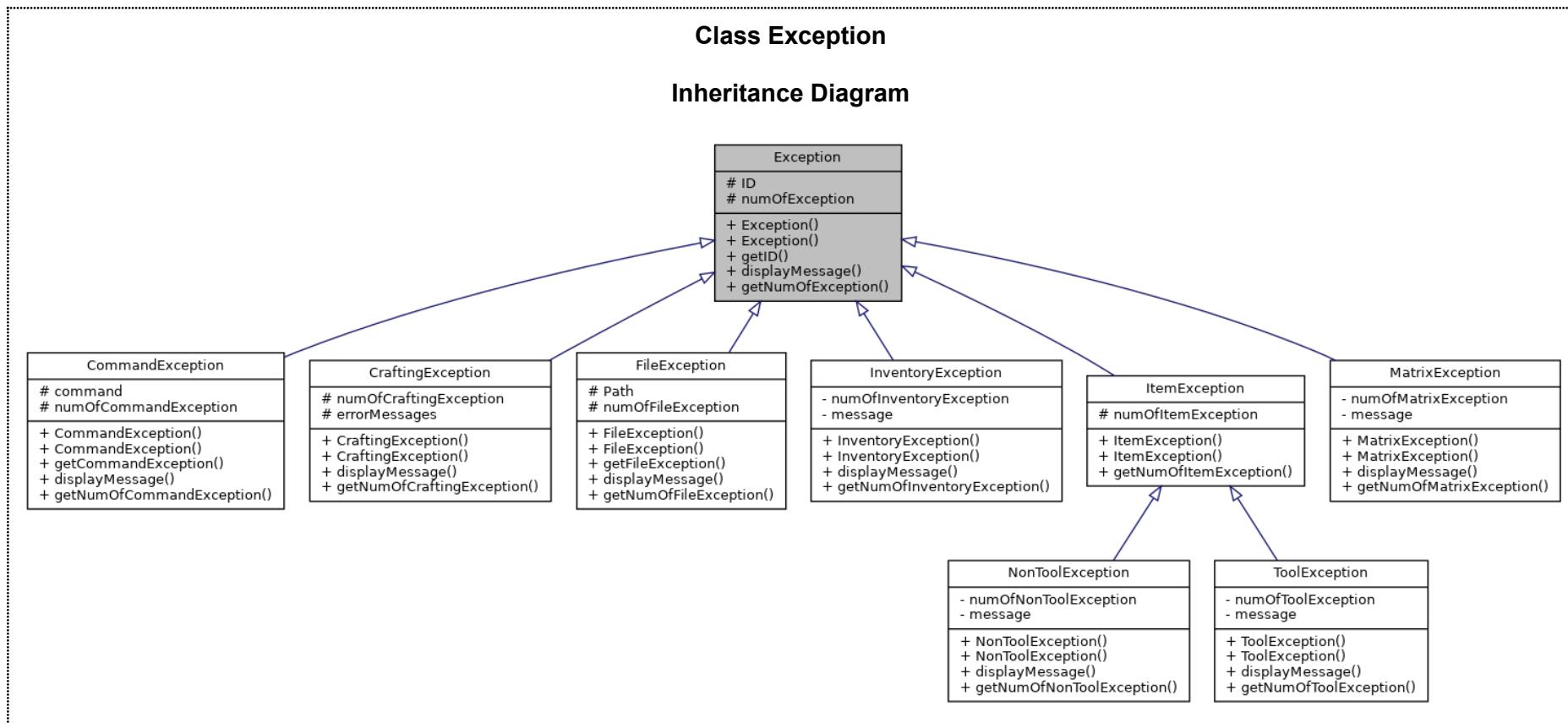
Class Matrix**Inheritance Diagram**

< Item * > < string >

Class MatrixException**Inheritance Diagram****Class NonToolException****Inheritance Diagram**

Class ToolException**Inheritance Diagram****Class ItemException****Inheritance Diagram****Class FileException****Inheritance Diagram**

Class CommandException**Inheritance Diagram****Class InventoryException****Inheritance Diagram****Class CraftingException****Inheritance Diagram**



c. Class Design

i. Item

Kelas Item banyak menerapkan konsep inheritance, polymorphism, operator overloading, dan function overloading dalam implementasinya. Inheritance digunakan untuk membuat jenis Item yang lebih spesifik, yaitu Tool dan Nontool. Polymorphism digunakan untuk menggeneralisasi tipe objek pada kelas anak yang diinstansiasi pada kelas-kelas lainnya. Kitacukup mendeklarasikan objek dengan tipe Item, tetapi objek tersebut dapat berperilaku baik seperti Tool maupun Nontool.

Kelebihan dari desain kelas Item ini adalah kita dapat dengan mudah membuat tipe collection of Item pada Inventory maupun CraftingTable. Kita tidak perlu membuat dua tipe collection yang berbeda-beda untuk setiap jenis Item yang diinstansiasi oleh pengguna. Kekurangan dari desain kelas Item ini adalah rawannya pengembalian metode secara abnormal akibat dilemparkannya exception. Terdapat beberapa metode yang ketika digunakan secara tidak teliti akan melemparkan exception. Oleh karena itu, dibutuhkan exception handling yang cukup kompleks untuk menggunakan kelas Item ini. Operator dan function overloading digunakan pula pada implementasi kelas Item untuk mempermudah kita untuk mengakses fungsionalitas kelas Item ini.

ii. **Inventory**

Kelas Inventory teragregasi dari beberapa kelas lain seperti Kelas Matrix dan kelas Item, keunggulan dengan menggunakan agregasi adalah code menjadi re-usability. Re-usability memainkan peran penting dalam pengembangan program kami. Re-usability dapat mempersingkat waktu pengembangan program, memudahkan dalam memindahkan komponen dari Kelas satu ke Kelas lain. Kekurangannya adalah program yang kami buat menjadi dependen/saling bergantung, apabila ada kesalahan dari sebuah kelas, maka kelas lain yang menggunakan kelas tersebut akan terdampak. Kendala yang kami temukan seperti ada perubahan method dari kelas yang menjadi komposisi dari kelas lain yang menyebabkan perubahan input output yang harus disesuaikan kembali.

iii. **Crafting**

Kelas Crafting merupakan kelas yang mempunyai atribut komposit Kelas Recipe untuk menyimpan resep-resep crafting dan juga mempunyai atribut matriks slot item crafting table yang mengimplementasikan Kelas Matrix dan Kelas Item. Slot item tersebut diimplementasikan layaknya slot item pada Kelas Inventory. Sehingga, antara Kelas Crafting dan Kelas Inventory mempunyai beberapa similaritas pada *getter* dan *setter*-nya, yang mana keduanya sama-sama memanfaatkan fungsi *slot-based getter* dan *setter* yang terdapat pada Kelas Matrix. Pada Kelas Crafting, pengurangan *code redundancy* dilakukan dengan menerapkan metode dan atribut pada kelas lain yang *reusable* pada Kelas Crafting sehingga rancangan dari program bersifat modular dan cenderung lebih efisien ketimbang harus mengimplementasikan fungsi yang sudah ada. Namun, hal ini membuat program saling bergantung satu sama lain sehingga pembuatan kelas bergantung dengan progres dari kelas lainnya. Untuk menangani ini, dibutuhkan *unit testing* demi mengurangi kemungkinan adanya kerusakan antarkelas yang diakibatkan oleh satu kelas yang terkait oleh beberapa kelas lainnya.

iv. **ConsoleIO dan Command**

Kelas ConsoleIO merupakan *child* dari kelas Command. Kelas Command itu sendiri dibuat untuk memanggil *method* yang telah diterapkan pada kelas lainnya untuk dapat digunakan di program utama. ConsoleIO sebagai *child* dari Command akan mampu mengakses *method* pada Command sehingga struktur dari program utama dibuat di kelas ini. Tujuan dibuatnya kelas ini agar pada file main.cpp tidak redundant dalam algoritmanya, dengan cukup memanggil *method* dari ConsoleIO akan dapat menjalankan program utamanya. Kelas Command juga dibuat agar penambahan *command* yang dapat dipakai menjadi lebih mudah karena cukup ditambahkan di kelas tersebut saja.

v. **FileIO**

Kelas FileIO banyak menggunakan C++ STL dalam implementasinya. Tujuan dibuatnya kelas ini adalah agar semua urusan Input/Output baik dari atau mengarah ke File diterapkan dalam kelas ini sehingga mudah untuk digunakan pada kelas lain. Salah satu contohnya adalah method dari FileIO listOfTool dan listOfNonTool akan mengembalikan vector berisi Tool dan NonTool yang akan digunakan pada kelas Inventory dan juga listOfRecipe akan digunakan pada kelas CraftingTable.

vi. **Matrix**

Kelas Matrix menggunakan konsep template, kelas generik, serta operator overloading dalam implementasinya. Kami menggunakan konsep kelas generik untuk mengabstraksi tipe data yang diagregasi pada kelas Matrix ini. Sebagai contoh, kelas Matrix ini diinstansiasi menjadi beberapa objek, dengan tipe data template digantikan dengan string serta pointer to Item. Selain itu, kelas Matrix juga menggunakan konsep operator overloading yang digunakan sebagai metode getter dan setter elemen. Kelebihan dari implementasi dengan cara tersebut adalah objek yang terinstansiasi pada kelas lain dapat dengan mudah menggunakan kelas Matrix ini tanpa perlu membaca bagian body kelas Matrik terlebih dahulu. Kekurangan dari implementasi dengan cara tersebut adalah rawan terjadinya error saat compile akibat adanya kelas template tersebut.

vii. **Exception**

Kelas Exception banyak menerapkan konsep inheritance dan polymorphism dalam implementasinya. Sama seperti kelas Item, kelas Exception menggunakan konsep inheritance dan polymorphism untuk menggeneralisir tipe objek pada kelas anak yang diinstansiasi dengan cukup mendeklarasikan tipe kelas dasarnya. Polymorphism juga berfungsi pada Exception handling, dimana kita harus melakukan catch Exception berdasarkan tipenya. Akan lebih mudah dan sederhana apabila kita cukup

menangkap Exception pada kelas dasarnya, dikarenakan dengan seperti itu kita tidak perlu mencantumkan seluruh kemungkinan tipe Exception yang dilemparkan oleh modul. Kelebihan dari implementasi dengan cara tersebut adalah mudahnya kelas Exception ini untuk di-inherit oleh kelas Exception yang lebih spesifik penggunaannya (seperti ItemException). Selain itu, objek dengan tipe Exception tersebut mudah digunakan pada try catch block dikarenakan sifat polymorphismnya tersebut (hanya perlu menangkap tipe dasarnya saja jika terjadi error). Kekurangan dari implementasi dengan cara tersebut adalah message yang dihasilkan ketika terlemparnya Exception akan susah terlacak (susah menjadi error apa yang menghasilkan Exception tersebut).

2. Penerapan Konsep OOP

2.1. Inheritance & Polymorphism

Konsep Inheritance dan Polymorphism digunakan di beberapa kelas pada program. Kelas yang menggunakan Inheritance dan Polymorphism adalah Command (Parent) dan ConsoleIO(Child), Exception (Parent) dan banyak Child-nya, Item (Parent) dengan Tool(Child) dan NonTool (Child). Inheritance digunakan untuk mengabstraksikan kelas yang terbentuk serta pula memungkinkan kita untuk menggunakan polymorphism dalam pemrograman berorientasi objek ini. Polymorphism itu sendiri digunakan untuk mempermudah pengolahan suatu objek dengan hanya cukup menggunakan tipe kelas dasarnya saja. Kedua konsep tersebut memiliki peran penting dalam pengoperasian pada objek dari kelas Exception dan Item. Kelas Exception dan Item sama-sama memiliki tipe kelas anak yang beragam, dimana polymorphism dibutuhkan untuk mengabstraksi tipe deklarasi objek-objek menjadi tipe kelas dasarnya tersebut ketika digunakan pada suatu tipe data collection (seperti array, list, dan vector).

```
1 class Tool : public Item {
2     private:
3         static int numOfTool;
4
5     int durability;
6
7     public:
8         Tool(); // default ctor
9         Tool(int, string, string, int); // user-defined ctor
10        Tool(const Tool&); // cctor
11        ~Tool(); // dtor
12
13    Tool& operator=(const Tool&); // operator assignment
14
15    int getQuantity() const; // quantity getter
16    int getDurability() const; // durability getter
17    void setQuantity(int); // quantity setter
18    void setDurability(int); // durability setter
19
20    static int getNumOfTool(); // numOfTool getter
21
22    bool isTool();
23
24    Item& operator+=(int); // durability sum and assignment with integer argument
25    Item& operator-=(int); // durability subtraction and assignment with integer argument
26
27    Item& operator+=(const Item&); // durability sum and assignment with Item argument
28    Item& operator-=(const Item&); // durability subtraction and assignment with Item argument
29
30    void input(istream&); // input method (delegated from stream)
31    void output(ostream&) const; // output method (delegated from stream)
32
33    void use(); // use item
34    void display(); // display tool
35};
```

```
1 class NonTool : public Item {
2     private:
3         static int numofNonTool;
4
5         int quantity;
6
7     public:
8         NonTool(); // default ctor
9         NonTool(int, string, string, int); // user-defined ctor
10        NonTool(const NonTool&); // cctor
11        ~NonTool(); // dtor
12
13    NonTool& operator=(const NonTool&); // assignment operator
14
15    int getQuantity() const; // quantity getter
16    int getDurability() const; // durability getter
17    void setQuantity(int); // quantity setter
18    void setDurability(int); // durability setter
19
20    static int getNumOfNonTool(); // numofNonTool getter
21
22    bool isTool();
23
24    Item& operator+=(int); // quantity sum and assignment with integer argument
25    Item& operator-=(int); // quantity subtraction and assignment with integer argument
26
27    Item& operator+=(const Item&); // quantity sum and assignment with Item argument
28    Item& operator-=(const Item&); // quantity subtraction and assignment with Item argument
29
30    void input(istream&); // input method (delegated from stream)
31    void output(ostream&) const; // output method (delegated from stream)
32
33    void use(); // use item
34    void display(); // display
35};
```

```
● ● ●
1 class Item {
2     protected:
3         static int numOfItem;
4
5         int ID;
6         string name;
7         string type;
8
9     public:
10        Item(); // default ctor
11        Item(int, string, string); // user-defined ctor
12        Item(const Item&); // ctor
13        virtual ~Item(); // virtual dtor
14
15        Item& operator=(const Item&); // assignment operator
16
17        int getID() const; // ID getter
18        void setID(int); // ID setter
19
20        string getName() const; // name getter
21        void setName(string); // name setter
22
23        string getType() const; // type getter
24        void setType(string); // type setter
25
26        virtual int getQuantity() const = 0; // quantity getter
27        virtual int getDurability() const = 0; // durability getter
28
29        virtual void setQuantity(int) = 0; // quantity setter
30        virtual void setDurability(int) = 0; // durability setter
31
32        static int getNumOfItem(); // numOfItem getter
33
34        friend istream& operator>>(istream&, Item&); // input stream
35        friend ostream& operator<<(ostream&, const Item&); // output stream
36
37        virtual bool isTool() = 0; // isItemTool getter
38
39        virtual Item& operator+=(int) = 0; // sum and assignment with integer argument
40        virtual Item& operator-=(int) = 0; // subtraction and assignment with integer argument
41
42        virtual Item& operator+=(const Item&) = 0; // sum and assignment with Item argument
43        virtual Item& operator-=(const Item&) = 0; // subtraction and assignment with Item argument
44
45        virtual void input(istream&) = 0; // input method (delegated from stream)
46        virtual void output(ostream&) const = 0; // output method (delegated from stream)
47
48        virtual void use() = 0; // use item
49        virtual void display(); // display item
50    };
}
```

2.2. Method/Operator Overloading

Konsep Method Overloading digunakan di beberapa kelas pada program yaitu CraftingTable dan ConsoleIO dan Operator Overloading digunakan pada MatrixInterface, Item, Tool, dan Nontool. Secara umum, *method overloading* diimplementasikan ketika terdapat beberapa fungsi berbeda yang memiliki logika penggunaan yang sama. Pembedaan antarfungsi dilakukan dengan membedakan parameter yang diterimanya. Sebagai contoh, pada *class CraftingTable* terdapat fungsi yang memindahkan *item* dari *crafting table* ke *inventory* dan ada juga fungsi yang melakukan sebaliknya dengan logika penggunaan yang mirip sehingga pemanggilannya dapat dibedakan hanya dengan membedakan parameter. *Operator overloading* digunakan untuk mempermudah penulisan. Contoh, pengambilan elemen pada *buffer* dari *matrix* berdasarkan *slotID* seharusnya dilakukan dengan mengubah *slotID* menjadi angka baris dan kolom terlebih dahulu dan kemudian mengakses *matrix->buffer[i][j]* dari matriks tersebut dengan i berupa baris dan j berupa kolom yang didapat dari *slotID*. Akan tetapi, penulisan dapat dipersingkat sehingga hanya perlu menulis *matrix[slotID]* dengan menggunakan operator[] yang merupakan *operator overloading*.

```
1 T& operator()(int, int) const; // index-based element getter
2 T& operator[](string) const; // slotID-based element getter
3
4 Matrix<T>& operator=(const Matrix<T>&); // assignment operator
```

```
1 // move item from inv to crafting table
2 void move(Inventory&, string, int, vector<string>);
3 // return item from crafting table to inv
4 void move(Inventory&, string, int, string);
```

```
1 Item& operator+=(int); // durability sum and assignment with integer argument
2 Item& operator-=(int); // durability subtraction and assignment with integer argument
3
4 Item& operator+=(const Item&); // durability sum and assignment with Item argument
5 Item& operator-=(const Item&); // durability subtra
```

```
1 // setter
2 void setCommand(string);
3 // override function for CLI-based command
4 void setCommand();
```

```
● ● ●  
1 Item& operator+=(int); // quantity sum and assignment with integer argument  
2 Item& operator-=(int); // quantity subtraction and assignment with integer argument  
3  
4 Item& operator+=(const Item&); // quantity sum and assignment with Item argument  
5 Item& operator-=(const Item&); // quantity subtraction and assignment with Item argument
```

2.3. Template & Generic Classes

Konsep Template dan Generic Classes digunakan pada kelas Matrix, CraftingTable, Recipe, dan Inventory. Dengan adanya template kelas generik, tidak perlu mendefinisikan kembali kelas Matrix ketika tipe elemen yang dibutuhkan oleh kelas yang berbeda-beda untuk setiap kebutuhan. Selain itu, kita juga dapat melakukan abstraksi terhadap atribut dan metode yang dimiliki oleh kelas generik tersebut dengan menggunakan tipe yang disediakan oleh template. Umumnya, kelas generik digunakan untuk mendefinisikan kelas dasar dengan tipe collection (kelas aggregat objek yang memiliki tipe sama). Pada program, kelas Matrix itu sendiri digunakan untuk merepresentasikan Matrix of string dan juga Matrix of pointer to Item. Meskipun kedua tipe kelas tersebut tidak memiliki kesamaan yang relevan, tetapi kita masih dapat menggunakan kedua tipe kelas tersebut sebagai tipe kelas template pada kelas generik Matrix.

```

1 template<class T>
2 class Matrix {
3     private:
4         int N, M;
5         T **buffer;
6
7     public:
8         Matrix(); // default ctor
9         Matrix(int, int); // user-defined ctor
10        Matrix(const Matrix&); // cctor
11        ~Matrix(); // dtor
12
13        int getLength() const; // N getter
14        int getWidth() const; // M getter
15        T& operator()(int, int) const; // index-based element getter
16        T& operator[](string) const; // slotID-based element getter
17
18        void setElmt(int, int, T); // index-based element setter
19        void setElmt(string, T); // slotID-based element setter
20
21        Matrix<T>& operator=(const Matrix<T>&); // assignment operator
22
23    };

```

```

1 class CraftingTable {
2     private:
3         // attributes
4         Recipe* recipes;
5         int NumOfRecipes;
6         int MaxRecipes;
7         const int row = 3;
8         const int col = 3;
9         Matrix<Item*> table;

```

```

1 class Recipe {
2     private:
3         // attributes
4         int row;
5         int col;
6         int productQty;
7         string productName;
8         Matrix<string> configRecipe;

```

```

1 class Inventory {
2     private:
3         static const int length = 3;
4         static const int width = 9;
5
6         // create Matrix of pointers to items
7         Matrix<Item*> table = Matrix<Item*>(length, width);

```

2.4. Exception

Konsep Exception hampir digunakan di setiap kelas karena banyak kasus yang perlu ditangani sehingga perlu dilakukan Exception. Seringkali, kita membutuhkan penanganan kesalahan yang muncul saat eksekusi program. Penanganan kesalahan tersebut membutuhkan kita untuk menambahkan instruksi-instruksi tertentu yang membuat program menjadi rumit dan rawan terjadinya ketidaktelitian pada implementasi. Oleh karena itu, C++ menyediakan fitur exception untuk menangani kesalahan yang terjadi saat runtime menggunakan sintaks throw, try, dan catch. Sintaks throw memiliki kegunaan yang mirip dengan sintaks return, dimana throw sendiri dapat dikatakan sebagai return dengan exception. Tetapi jika suatu method selesai dengan throw, maka method tersebut dikatakan selesai secara abnormal. Ketika kita ingin

mengakses exception yang di-throw oleh suatu method, maka block method tersebut harus terdapat di dalam sintaks try, serta memiliki sintaks catch untuk menangkap objek exception yang di-throw.

```
● ● ●
1 class Exception {
2     protected:
3         static int numOfException;
4         const int ID;
5
6     public:
7         // tidak memerlukan default ctor (tidak ada list of exception), assignment operator (tidak ada assignment), dan dtor (perlunya perhitungan jumlah object Exception setelah dihapus)
8         Exception(int); // user-defined ctor
9         Exception(const Exception&); // cctor
10
11     static int getNumOfException(); // numOfException getter
12     int getID() const; // ID getter
13
14     virtual void displayMessage() const = 0; // message display
15 }
```

```

1 // initilaize file IO
2 string line;
3 ifstream wordsConfigFile(path);
4
5 if (!wordsConfigFile)
6 {
7     throw FileException(path);
8 }
9 else
10 {
11     // read each line
12     while (getline(wordsConfigFile, line))
13     {
14         words = wordsFromSentence(line);
15         // IO each line to vector of words
16         vectorOfWords.push_back(words);
17     }
}

```

```

1 void Inventory::use(const string slotID){
2     if(this->table[slotID] == NULL){
3         throw InventoryException(5);
4     }
5     try{
6         this->table[slotID]->use();
7         if(this->table[slotID]->getDurability()==0){
8             this->table[slotID] = NULL;
9         }
10    }
11    catch(ToolException &e){
12        e.displayMessage();
13    }
14 }

```

```

1 void Inventory::moveInInventory(string slotSrc, int qty, string slotTarget){
2     // slot asal tidak boleh kosong
3     if(this->table[slotSrc] == NULL){
4         throw InventoryException(5);
5     }
6     //jika slot tujuan tidak kosong maka id harus sama
7     if(this->table[slotTarget] != NULL){
8         if (this->table[slotSrc]->getID() != this->table[slotTarget]->getID()){
9             throw InventoryException(2);
10        }
11    }
12    //jika slot src adalah tool maka tidak boleh dipindah ke selain null
13    if(this->table[slotSrc]->isTool()){
14        if(this->table[slotTarget] != NULL){
15            throw InventoryException(1);
16        }
17    }
18    if(qty != 1){
19        throw InventoryException(6);
20    }
}

```

2.5. C++ Standard Template Library

C++ STL cukup banyak digunakan pada setiap kelas, salah satu library yang paling banyak digunakan pada setiap kelas adalah `<string>` karena cukup banyak menyimpan atribut dengan type string. Library lain yang digunakan adalah `<iostream>` dan `<fstream>` untuk input dan output ke file, dan juga `<filesystem>` pada FileIO. Penggunaan STL bertujuan untuk menggunakan fungsi-fungsi yang tersedia dalam library STL sehingga tidak perlu diimplementasikan sendiri. Selain itu, kami juga menggunakan STL vector sebagai tipe collection yang mudah digunakan dan diubah ukurannya.

```
1 #ifndef FILE_IO_INTERFACE_HPP
2 #define FILE_IO_INTERFACE_HPP
3
4 #include "../item/itemInterface.hpp"
5 #include "../item/nonToolInterface.hpp"
6 #include "../item/toolInterface.hpp"
7 #include "../crafting/recipeInterface.hpp"
8 #include "../exception/fileExceptionInterface.hpp"
9 #include "string.h"
10 #include <filesystem>
11 #include <fstream>
12 #include <iostream>
13 #include <string>
14 #include <vector>
15
16 class FileIO
17 {
18 protected:
19     static string configPath;
20     static string testsPath;
21
22 public:
23     // getter for configPath
24     static string getConfigPath();
25     // getter for testsPath
26     static string getTestsPath();
27     // get count of test file
28     int countOfTestFile(string path);
29     // split words from sentences to a vector of string
30     vector<string> wordsFromSentence(string);
31     // read each words from file
32     vector<vector<string>> vectorOfWords(string);
33     // vector of tools
34     vector<Tool> listOfTool();
35     // vector of non tools
36     vector<NonTool> listOfNonTool();
37     // vector of recipe
38     vector<Recipe> listOfRecipe();
39     // main program
40     // void start(string);
41     // display tool
42     void displayTool();
43     // display non tool
44     void displayNonTool();
45     // display recipe
46     void displayRecipe();
47 };
48
49 #endif
```

2.6. Konsep OOP lain

Konsep OOP lain yang digunakan dalam program adalah Abstract Base Class, Collection, dan Composition. Konsep Abstract Base Class diterapkan pada base class Item yang di-inherit kepada kelas Tool dan NonTool untuk di-implementasi sesuai dengan kelasnya. Konsep Composition digunakan pada kelas CraftingTable yang pada atributnya menggunakan kelas Recipe. Konsep Collection digunakan pada pembuatan kelas Matrix dimana atribut *buffer* pada kelas tersebut berbentuk array of array of object.

```

● ● ●
1 class Item {
2     protected:
3         static int numOfItem;
4
5         int ID;
6         string name;
7         string type;
8
9     public:
10        Item(); // default ctor
11        Item(int, string, string); // user-defined ctor
12        Item(const Item&); // ctor
13        virtual ~Item(); // virtual dtor
14
15        Item& operator=(const Item&); // assignment operator
16
17        int getID() const; // ID getter
18        void setID(int); // ID setter
19
20        string getName() const; // name getter
21        void setName(string); // name setter
22
23        string getType() const; // type getter
24        void setType(string); // type setter
25
26        virtual int getQuantity() const = 0; // quantity getter
27        virtual int getDurability() const = 0; // durability getter
28
29        virtual void setQuantity(int) = 0; // quantity setter
30        virtual void setDurability(int) = 0; // durability setter
31
32        static int getNumOfItem(); // numOfItem getter
33
34        friend istream& operator>>(istream&, Item&); // input stream
35        friend ostream& operator<<(ostream&, const Item&); // output stream
36

```

```

36
37         virtual bool isTool() = 0; // isItemTool getter
38
39         virtual Item& operator+=(int) = 0; // sum and assignment with integer argument
40         virtual Item& operator-=(int) = 0; // subtraction and assignment with integer argument
41
42         virtual Item& operator+=(const Item&) = 0; // sum and assignment with Item argument
43         virtual Item& operator-=(const Item&) = 0; // subtraction and assignment with Item argument
44
45         virtual void input(istream&) = 0; // input method (delegated from stream)
46         virtual void output(ostream&) const = 0; // output method (delegated from stream)
47
48         virtual void use() = 0; // use item
49         virtual void display(); // display item
50     };

```

```

● ● ●
1 class CraftingTable {
2     private:
3         // attributes
4         Recipe* recipes;
5         int NumOfRecipes;
6         int MaxRecipes;
7         const int row = 3;
8         const int col = 3;
9         Matrix<Item*> table;

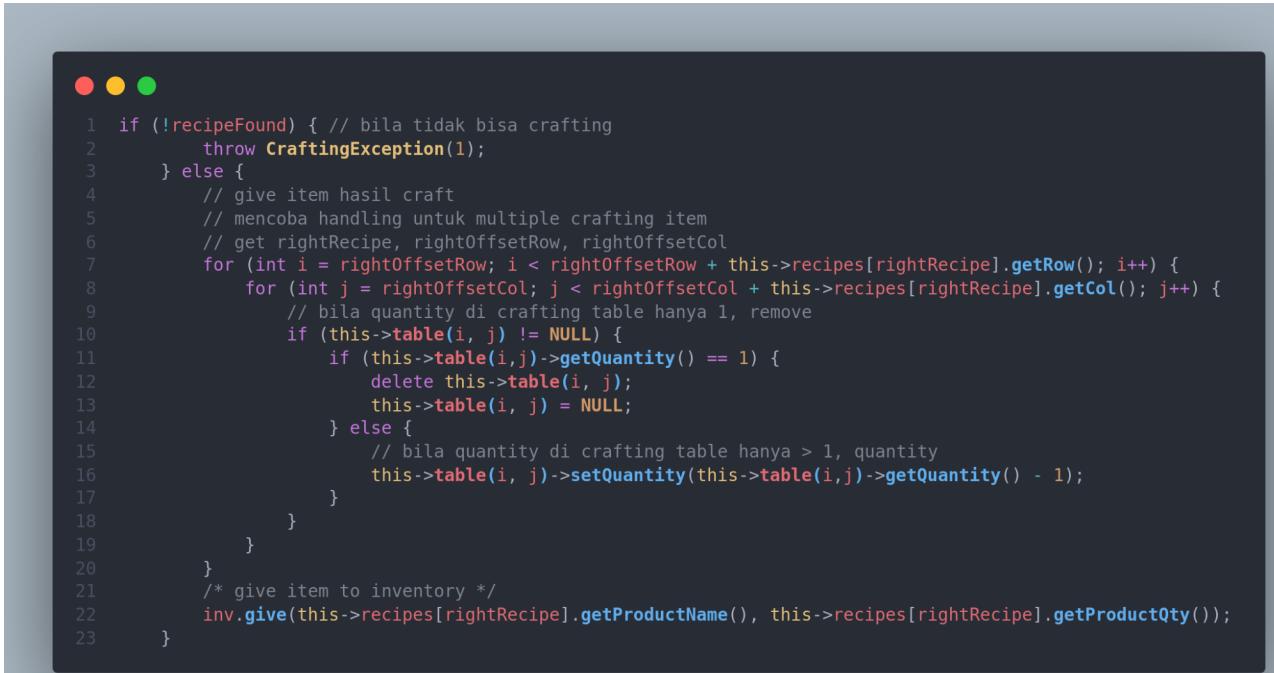
```

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Multiple Crafting

Bonus *multiple crafting* mempengaruhi bagaimana setiap slot pada crafting table memegang setiap item nontool, namun tidak pada slot dengan item tool yang hanya dapat berkuantitas satu. Yang sebelumnya pada spesifikasi setiap item nontool pada slot crafting table hanya dapat berjumlah satu berubah sehingga berjumlah lebih dari satu.



```

1 if (!recipeFound) { // bila tidak bisa crafting
2     throw CraftingException(1);
3 } else {
4     // give item hasil craft
5     // mencoba handling untuk multiple crafting item
6     // get rightRecipe, rightOffsetRow, rightOffsetCol
7     for (int i = rightOffsetRow; i < rightOffsetRow + this->recipes[rightRecipe].getRow(); i++) {
8         for (int j = rightOffsetCol; j < rightOffsetCol + this->recipes[rightRecipe].getCol(); j++) {
9             // bila quantity di crafting table hanya 1, remove
10            if (this->table(i, j) != NULL) {
11                if (this->table(i,j)->getQuantity() == 1) {
12                    delete this->table(i, j);
13                    this->table(i, j) = NULL;
14                } else {
15                    // bila quantity di crafting table hanya > 1, quantity
16                    this->table(i, j)->setQuantity(this->table(i,j)->getQuantity() - 1);
17                }
18            }
19        }
20    }
21    /* give item to inventory */
22    inv.give(this->recipes[rightRecipe].getProductName(), this->recipes[rightRecipe].getProductQty());
23}

```

Implementasi multiple crafting ini berpengaruh pada bagaimana dilakukannya crafting dengan command CRAFT. Untuk setiap item pada slot crafting table yang digunakan dalam crafting, bila jumlah item pada slot crafting table yang digunakan untuk crafting lebih dari satu maka jumlah item tersebut dikurangi satu. Namun, bila jumlah item hanya satu, maka item pada slot tersebut dihapus dan di-dealokasi. Selain itu, CRAFT akan mengulang hingga pada akhirnya crafting table kosong, penggabungan dua tool tidak memungkinkan, atau tidak ada resep yang cocok.



```
● ● ●
1 // mengisi slot crafting untuk non-tool
2 else {
3     for (int i = 0; i < n; i++) {
4         Item* destSlot = this->getElmt(slotCrafts[i]);
5         if (destSlot != NULL) {
6             if (!destSlot->isTool()) {
7                 if ((destSlot->getName() != item->getName() || destSlot->getType() != item->getType())) {
8                     throw CraftingException(3);
9                 }
10                else {
11                    inSlotDest = destSlot->getQuantity();
12                }
13            }
14        }
15        this->setElmt(slotCrafts[i], (new NonTool(item->getID(), item->getName(), item->getType(), inSlotDest + toGive)));
16        inSlotDest = 0;
17    }
18 }
```



```
● ● ●
1 Item* destSlot = this->getElmt(slotCrafts[0]);
2         if (destSlot != NULL) {
3             if (destSlot->isTool()) {
4                 throw CraftingException(3);
5             }
6             else {
7                 if ((destSlot->getName() != item->getName() || destSlot->getType() != item->getType())) {
8                     throw CraftingException(3);
9                 }
10                else {
11                    inSlotDest = destSlot->getQuantity();
12                }
13            }
14        }
```



```

1 Item* destSlot = inv.getElmt(slotInv);
2 if (destSlot != NULL) {
3     if (destSlot->isTool()) {
4         throw CraftingException(3);
5     }
6     else {
7         if ((destSlot->getName() != item->getName() || destSlot->getType() != item->getType())) {
8             throw CraftingException(3);
9         }
10        else {
11            inSlotDest = destSlot->getQuantity();
12        }
13    }
14}

```

Selain itu, implementasi multiple crafting juga berpengaruh pada pemindahan item nontool dari inventory ke crafting table sehingga tidak ada batasan jumlah item sebanyak satu pada slot crafting table. Perlu dicek terlebih dahulu apakah slot pada crafting table sudah kosong atau tidak. Bila kosong maka item nontool dengan kuantitas tertentu dapat langsung dipindahkan. Namun bila tidak, maka perlu dicek apakah item yang dipindah adalah item yang sama dengan yang sudah ada di slot crafting table.

3.1.2. Item dan Tool Baru

Penambahan Item dan Tool baru dilakukan dengan cara merubah folder config. Untuk penambahan Item (Tool dan Nontool) dapat langsung ditambahkan pada file.txt dengan menuliskan ID Item, Nama Item, Tipe, dan Kategori (Tool/NonTool). Jika ingin menambahkan recipe juga bisa dilakukan dengan menambahkan nama recipe .txt yang diinginkan pada folder recipe lalu mengisi konfigurasi recipe baru tersebut. Ada banyak Item dan Recipe yang ditambahkan pada program ini.

List Item (yang ditambahkan): 25 FLINT - NONTOP 26 FLINT_AND_STEEL - TOOL	DIAMOND_HOEK.txt FISHING_ROD.txt FLINT_AND_STEEL.txt
---	--

27 GOLD_INGOT - NONTOOL	GOLD_AXE.txt
28 GOLD_NUGGET - NONTOOL	GOLD_HOEK.txt
29 GOLD_PICKAXE - TOOL	GOLD_INGOT.txt
30 GOLD_AXE - TOOL	GOLD_NUGGET.txt
31 GOLD_SWORD - TOOL	GOLD_PICKAXE.txt
32 WOODEN_HOEK - TOOL	GOLD_SWORD.txt
33 IRON_HOEK - TOOL	IRON_HOEK.txt
34 GOLD_HOEK - TOOL	LADDER.txt
35 DIAMOND_HOEK - TOOL	PIG.txt
36 SOMBERSTONE STONE NONTOOL	SITE_OF_GRACE.txt
37 LADDER - NONTOOL	SLIME_BLOCK.txt
38 SLIME_BALL - NONTOOL	STONE_HOEK.txt
39 SLIME_BLOCK - NONTOOL	VILLAGER.txt
40 GRACE - NONTOOL	WOODEN_HOEK.txt
41 SITE_OF_GRACE - NONTOOL	
42 STRING - NONTOOL	
43 FISHING_ROD - TOOL	
44 PIG - TOOL	
45 PIG_HEAD - NONTOOL	
46 VILLAGER - TOOL	
47 VILLAGER_HEAD - NONTOOL	
48 STONE_HOEK - TOOL	

3.1.3. Unit Testing Implementation

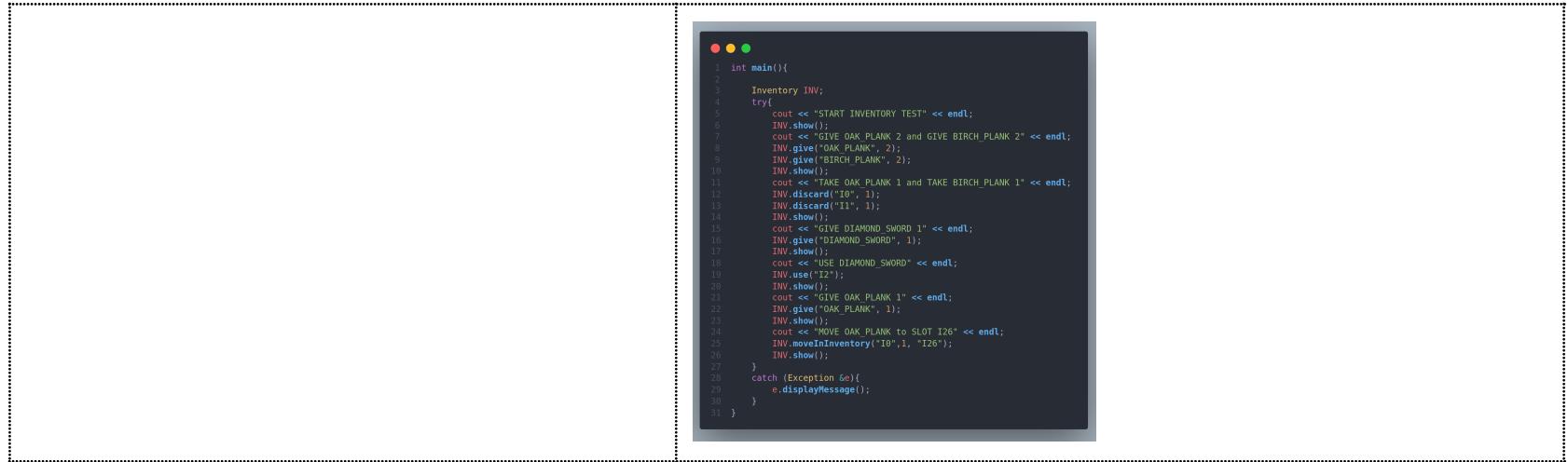
Unit testing dibutuhkan sebagai sarana untuk memastikan bahwa masing-masing komponen dari program dapat berjalan dengan baik sehingga meminimalisir kerusakan keseluruhan ketika seluruh komponen diintegrasikan. Oleh karena itu, kelompok kami mengimplementasikan dengan cara membuat *driver* untuk 4 *class* yang saling terintegrasi yakni *driverCrafting*, *driverInventory*, *driverIO*, dan *driverItem* seperti pada *screenshot* di bawah untuk menjamin keberfungsiannya dari metode yang terdapat pada *class* terkait.

```

1 int main()
2 {
3     // file IO
4     FileIO FIO;
5
6     // display item
7     FIO.displayNonTool();
8     FIO.displayTool();
9     // display recipe
10    FIO.displayRecipe();
11
12    // get config path
13    cout << "config path: " << FIO.getConfigPath() << endl;
14    cout << "tests path : " << FIO.getTestsPath() << endl;
15
16    // get all line from a file
17    try
18    {
19        string filePath = FIO.getTestsPath() + "/1.in";
20        vector<string> testFile = FIO.vectorOfWords(filePath);
21
22        // print all string from testfile
23        for (int i = 0; i < testFile.size(); i++)
24        {
25            for (int j = 0; j < testFile[i].size(); j++)
26            {
27                cout << testFile[i][j] << " ";
28            }
29            cout << endl;
30        }
31    }
32    catch (FileNotFoundException &FE)
33    {
34        FE.displayMessage();
35    }
36
37    // get all line from a non existent file
38    try
39    {
40        string errorPath = FIO.getTestsPath() + "/0.in";
41        vector<string> testError = FIO.vectorOfWords(errorPath);
42    }
43    catch (FileNotFoundException &FE)
44    {
45        FE.displayMessage();
46    }
47
48    // command and console IO
49    ConsoleIO CIO;
50    Inventory Inv = Inventory();
51    CraftingTable Craft = CraftingTable();
52
53    // manually set command
54    CIO.setCommand("SHOW");
55    CIO.SHOW(Inv, Craft);
56
57    // print num of command used (should be 1)
58    cout << "Number of Command used: " << CIO.getNumOfCommand() << endl;
59
60    // override to ask from console
61    cout << "Set your command: ";
62    CIO.setCommand();
63
64    // print command
65    cout << "Command from user: " << CIO.getCommand() << endl;
66
67    // start program
68    cout << endl;
69    cout << "start of the program" << endl;
70    cout << "type EXIT to close the program" << endl;
71    CIO.start();
72
73    cout << "you reached the end of the program" << endl;
74    cout << "Number of valid Command used: " << CIO.getNumOfCommand() << endl;
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
653
654
655
655
656
657
657
658
659
659
660
661
661
662
662
663
663
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
15
```

```

1 int main() {
2
3     try {
4         Item *Oak_Log1 = new NonTool(1, "OAK_LOG", "LOG", 72); // THROW EXCEPTION
5         delete Oak_Log1;
6
7     } catch (ItemException &e) {
8         e.displayMessage();
9     }
10
11    try {
12        Item *Iron_Sword = new Tool(23, "IRON_SWORD", "-", 15);
13        delete Iron_Sword;
14
15    } catch (ItemException &e) {
16        e.displayMessage();
17    }
18
19    try {
20        Item *Oak_Log1 = new NonTool(1, "OAK_LOG", "LOG", 32);
21        Item *Oak_Log2 = new NonTool(1, "OAK_LOG", "LOG", 32);
22
23        *(Oak_Log1) += *(Oak_Log2);
24        cout << *Oak_Log1 << endl;
25
26        *(Oak_Log1) += *(Oak_Log2); // THROW EXCEPTION
27        cout << *Oak_Log1 << endl;
28
29        delete Oak_Log1;
30        delete Oak_Log2;
31
32    } catch (ItemException &e) {
33        e.displayMessage();
34    }
35
36    try {
37        Item *Wooden_Pickaxe = new Tool(13, "WOODEN_PICKAXE", "-", 10);
38        Item *Stick = new NonTool(7, "STICK", "-", 32);
39
40        *(Wooden_Pickaxe) += *(Stick); // THROW EXCEPTION
41
42        delete Wooden_Pickaxe;
43        delete Stick;
44
45    } catch (ItemException &e) {
46        e.displayMessage();
47    }
48
49    try {
50        Item *Diamond = new NonTool(12, "DIAMOND", "-", 64);
51
52        cout << Diamond->getQuantity() << endl;
53        cout << Diamond->getDurability() << endl; // THROW EXCEPTION
54
55        delete Diamond;
56
57    } catch (ItemException &e) {
58        e.displayMessage();
59    }
60
61    try {
62        Item *Diamond_Sword = new Tool(24, "DIAMOND_SWORD", "-", 5);
63
64        cout << Diamond_Sword->getDurability() << endl;
65        cout << Diamond_Sword->getQuantity() << endl; // THROW EXCEPTION
66
67        delete Diamond_Sword;
68
69    } catch (ItemException &e) {
70        e.displayMessage();
71    }
72
73    try {
74        Item *Diamond_Sword = new Tool(24, "DIAMOND_SWORD", "-", 5);
75        Item *Diamond = new NonTool(12, "DIAMOND", "-", 64);
76
77        Diamond_Sword->use();
78        cout << Diamond_Sword << endl;
79
80        Diamond_Sword->use();
81        cout << Diamond_Sword << endl;
82
83        Diamond_Sword->use();
84        cout << Diamond_Sword << endl;
85
86        Diamond->use(); // THROW EXCEPTION
87        cout << "Diamond" << endl;
88
89        delete Diamond_Sword;
90        delete Diamond;
91
92    } catch (ItemException &e) {
93        e.displayMessage();
94    }
95
96    try {
97        Item *Wooden_Pickaxe = new Tool(13, "WOODEN_PICKAXE", "-", 10);
98        Item *Stick = new NonTool(7, "STICK", "-", 32);
99
100       cout << Wooden_Pickaxe->isTool() << endl;
101       cout << Stick->isTool() << endl;
102
103       delete Wooden_Pickaxe;
104       delete Stick;
105
106    } catch (ItemException &e) {
107        e.displayMessage();
108    }
109
110 }
```



3.2. Bonus Kreasi Mandiri

3.2.1 PEEK

Kelompok kami memutuskan untuk menerapkan sebuah bonus kreasi berupa *command PEEK* yang diimplementasikan pada *recipe* yang terdapat pada *inventory*. Bonus ini menambahkan fitur yang dapat menampilkan resep dari suatu *craftable item* dengan memberikan *command PEEK <ITEM_NAME>/<ITEM_ID>* untuk melihat deskripsi item tersebut beserta baris, kolom, *product name*, *product quantity*, serta konfigurasi dari *item* tersebut apabila *item* tersebut *craftable*. Jika *item* tidak ditemukan maka tidak akan dilanjutkan untuk menampilkan resep dan akan mengeluarkan *exception "Item tidak tersedia"*. Apabila tidak ditemukan resep dengan

product name atau ID yang sesuai dengan *input* nama/ID *item*, akan dikeluarkan tulisan berupa “RECIPE NOT FOUND” untuk menjelaskan kepada pengguna bahwa resep tidak ditemukan.

```

1 void Inventory::peek(string itemName)
2 {
3     FileIO FIO;
4     vector<Tool> tools = FIO.listOfTool();
5     vector<NonTool> nonTools = FIO.listOfNonTool();
6     vector<Recipe> recipes = FIO.listOfRecipe();
7     bool itemFound = false;
8
9     // check for tool
10    cout << "Item: " << endl;
11    for (int i = 0; i < tools.size() && !itemFound; i++)
12    {
13        if (itemName == tools[i].getName())
14        {
15            tools[i].display();
16            cout << endl;
17            itemFound = true;
18        }
19    }
20
21    // check for nontool
22    for (int i = 0; i < nonTools.size() && !itemFound; i++)
23    {
24        if (itemName == nonTools[i].getName())
25        {
26            nonTools[i].display();
27            cout << endl;
28            itemFound = true;
29        }
30    }
31
32    // check if item is found
33    if (!itemFound)
34    {
35        for (int i = 0; i < recipes.size(); i++)
36        {
37            if (itemName == recipes[i].getProductName())
38            {
39                recipes[i].display();
40                cout << endl;
41                return;
42            }
43        }
44        throw InventoryException();
45    }
46    else
47    {
48        throw InventoryException(4);
49    }
50 }
```

PEEK SITE_OF_GRACE
 Item:
 ID : 41
 Name : SITE_OF_GRACE
 Type : -
 Quantity : 0
 Row : 3
 Col : 3
 Product Name : SITE_OF_GRACE
 Product Qty : 1
 Configuration:
 GRACE GRACE GRACE
 GRACE - GRACE
 GRACE GRACE GRACE

PEEK FIKRON
 Item:
 Item tidak terdaftar

3.2.2 Pewarnaan Pada SHOW

Kelompok kami menerapkan bonus pewarnaan setiap slot pada inventory dan crafting table agar dapat lebih membedakan item tool dan nontool. Ketika dilakukan command SHOW, slot yang berisi item tool diberi warna Bold Magenta, item nontool diberi warna Bold Cyan, dan slot yang kosong diberi warna default terminal. Pewarnaan teks pada slot dilakukan dengan string berisi kode warna ANSI escape code sesuai yang tersedia pada terminal Linux/Mac yang kemudian diberikan ke *cout*.

```

1 // show crafting table
2 void CraftingTable::show() const {
3     for (int i = 0; i < this->row; i++) {
4         cout << "\t\t\t\t\t\t";
5         for (int j = 0; j < this->col; j++) {
6             if (this->table(i, j) == NULL) {
7                 cout << "[C" << i*3+j << "|0]\t\t";
8                 if (j == 2) {
9                     cout << endl;
10                }
11            }
12            else {
13                if (this->table(i,j)->isTool()){
14                    cout << BOLDMAGENTA << "[C" << i*3+j << "|" << this->
15                    table(i, j)->getID() << "|" << this->table(i, j)->getDurability() << "]\t
16                    " << RESET;
17                    if (j == 2) {
18                        cout << endl;
19                    }
20                }
21                else{
22                    cout << BOLDCYAN << "[C" << i*3+j << "|" << this->tab
23                    le(i, j)->getID() << "|" << this->table(i, j)->getQuantity() << "]\t" <<
24                    RESET;
25                    if (j == 2) {
26                        cout << endl;
27                    }
28                }
29            cout << endl;
30        }
31    }
32 }
```

```

1 void Inventory::show() const{
2     for (int i = 0; i < length; i++) {
3         for (int j = 0; j < width; j++) {
4             if (this->table(i, j) == NULL) {
5                 cout << "[I" << i*9+j << "|0]\t\t";
6                 if (j == 8) {
7                     cout << endl;
8                 }
9             }
10            else {
11                if (this->table(i,j)->isTool()){
12                    cout << BOLDMAGENTA << "[I" << i*9+j << "|" << this-
13                    >table(i, j)->getID() << "|" << this->table(i, j)->getDurability() << "]"
14                    "\t" << RESET;
15                    if (j == 8) {
16                        cout << endl;
17                    }
18                }
19                else{
20                    cout << BOLDCYAN << "[I" << i*9+j << "|" << this->
21                    ble(i, j)->getID() << "|" << this->table(i, j)->getQuantity() << "]\t" <<
22                    RESET;
23                    if (j == 8) {
24                        cout << endl;
25                    }
26                }
27            }
28        }
29    }
30 }
```

```
● ● ●  
1 //the following are UBUNTU/LINUX, and MacOS ONLY terminal color codes.  
2 #define RESET    "\033[0m"  
3 #define BOLDMAGENTA "\033[1m\033[35m"      /* Bold Magenta */  
4 #define BOLDCYAN   "\033[1m\033[36m"      /* Bold Cyan */  
5
```

[I0 40 1]	[I1 24 10]	[I2 0]	[I3 0]	[I4 0]	[I5 0]	[I6 0]	[I7 0]	[I8 0]
[I9 0]	[I10 0]	[I11 0]	[I12 0]	[I13 0]	[I14 0]	[I15 0]	[I16 0]	[I17 0]
[I18 0]	[I19 0]	[I20 0]	[I21 0]	[I22 0]	[I23 0]	[I24 0]	[I25 0]	[I26 0]
			[C0 40 1]	[C1 24 10]	[C2 0]			
			[C3 0]	[C4 0]	[C5 0]			
			[C6 0]	[C7 0]	[C8 0]			
[I0 0]	[I1 0]	[I2 0]	[I3 0]	[I4 0]	[I5 0]	[I6 0]	[I7 0]	[I8 0]
[I9 0]	[I10 0]	[I11 0]	[I12 0]	[I13 0]	[I14 0]	[I15 0]	[I16 0]	[I17 0]
[I18 0]	[I19 0]	[I20 0]	[I21 0]	[I22 0]	[I23 0]	[I24 0]	[I25 0]	[I26 0]

4. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
Item	13520065	1352023, 13520044, 13520056, 13520065, 13520083
Inventory	13520056	1352023, 13520044, 13520056, 13520065, 13520083
Crafting	13520023, 13520083	1352023, 13520044, 13520056, 13520065, 13520083
ConsoleIO	13520044	1352023, 13520044, 13520056, 13520065, 13520083
FileIO	13520044	1352023, 13520044, 13520056, 13520065, 13520083
Exception	13520023, 13520044, 13520056, 13520065, 13520083	1352023, 13520044, 13520056, 13520065, 13520083
Matrix	13520065, 13520083	1352023, 13520044, 13520056, 13520065, 13520083