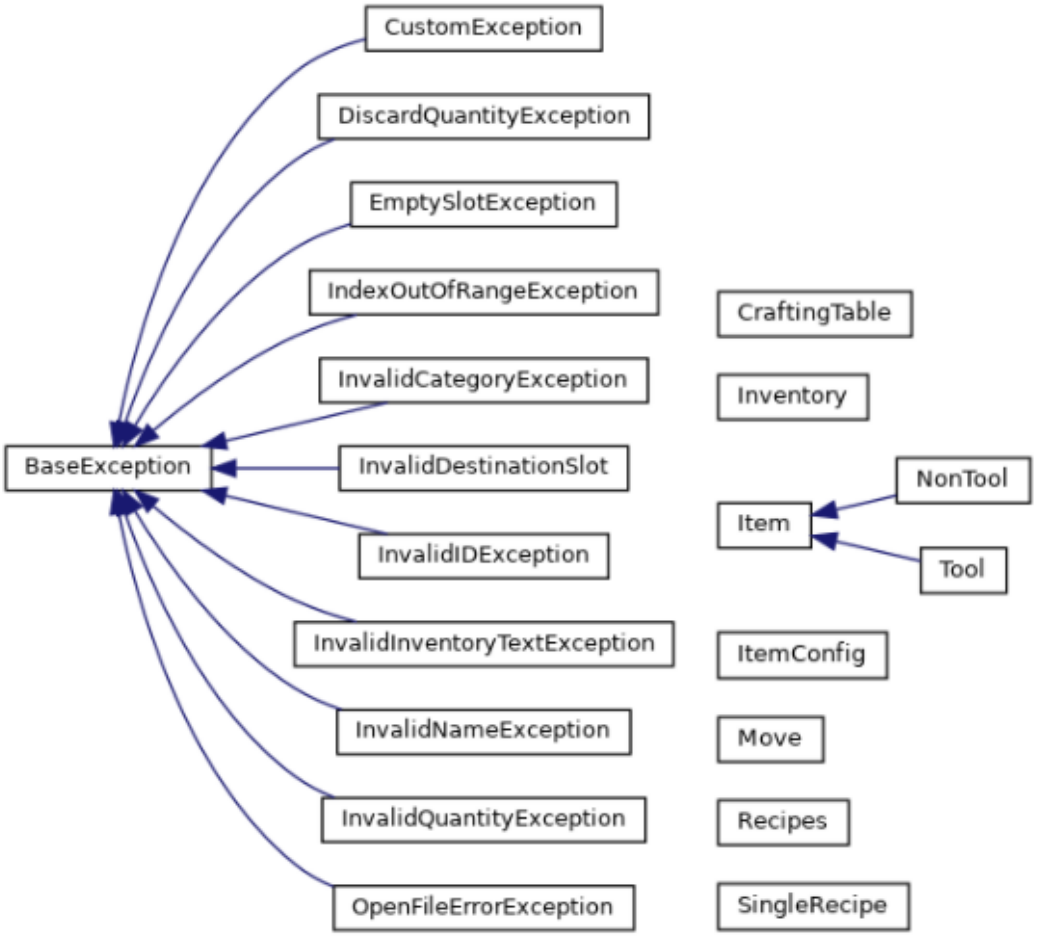
Kelas : 03
Nomor Kelompok : 01
Nama Kelompok : Pijat C++
1. 13520036 / I Gede Arya Raditya Parameswara
2. 13520048 / Arik Rayi Arkananta
3. 13520075 / Samuel Christopher Swandi
4. 13520078 / Grace Claudia
5. 13520081 / Andhika Arta Aryanto
6. 13520132 / Januar Budi Ghifari
Asisten Pembimbing: Jonathan Yudi Gunawan

1. Diagram Kelas

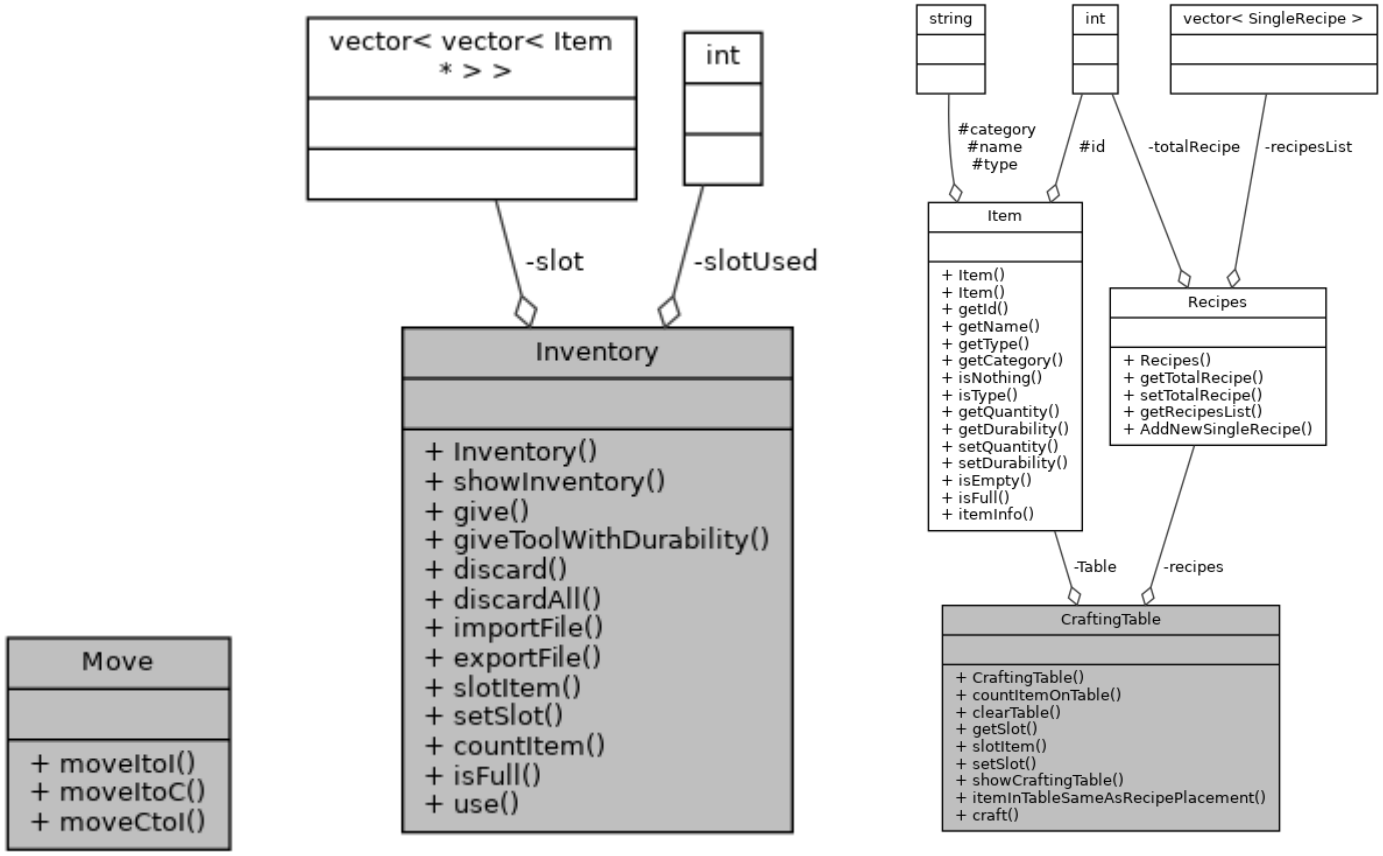
A. Hierarchy Diagram secara Umum

Hierarchy diagram adalah diagram yang menjelaskan masing - masing kelas , mendaftarkan atribut serta method tiap kelas, lalu menggambarkan apabila kelas tersebut memiliki *child class*

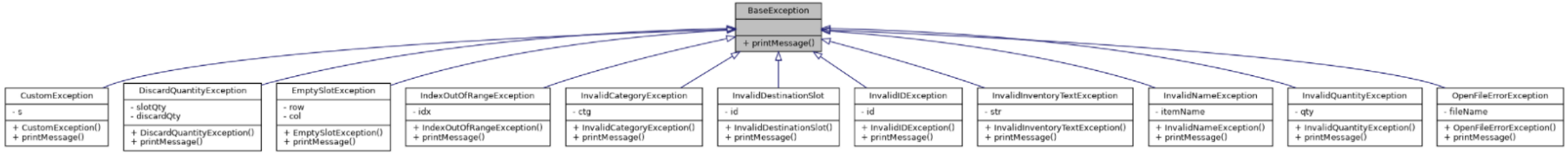


Gambar A.1 Hierarchy Diagram Program

B. Hierarchy Diagram Masing - Masing Kelas

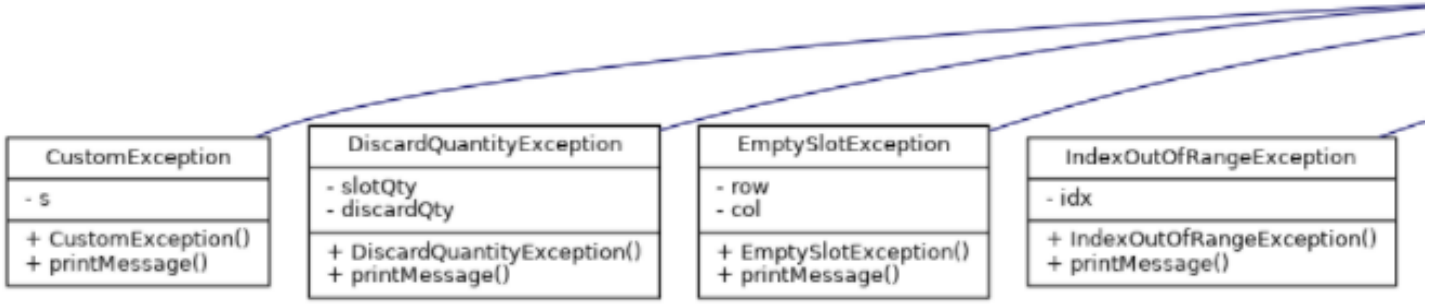


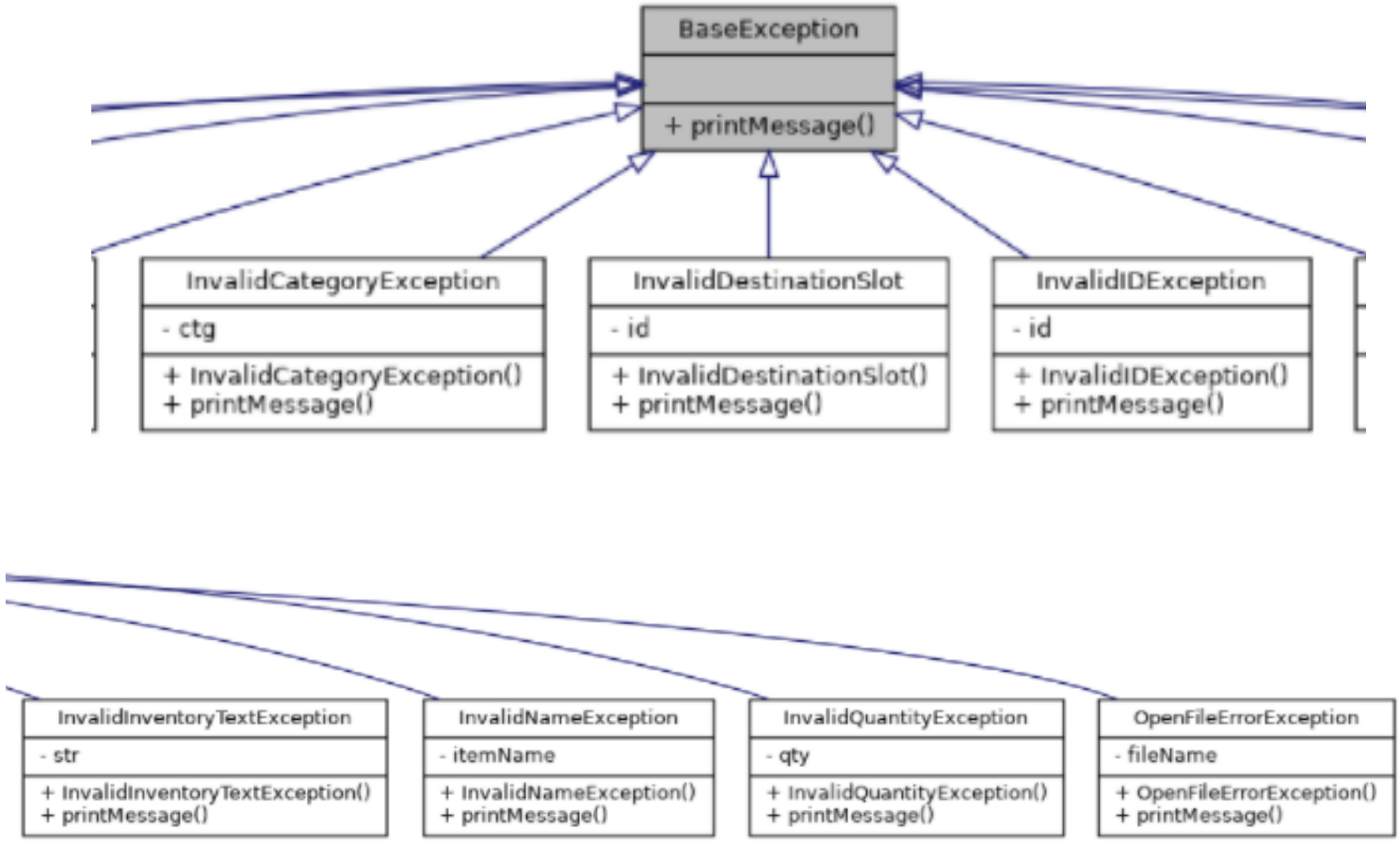
Gambar B.1 Hierarchy Diagram untuk Class Move, Inventory, dan CraftingTable

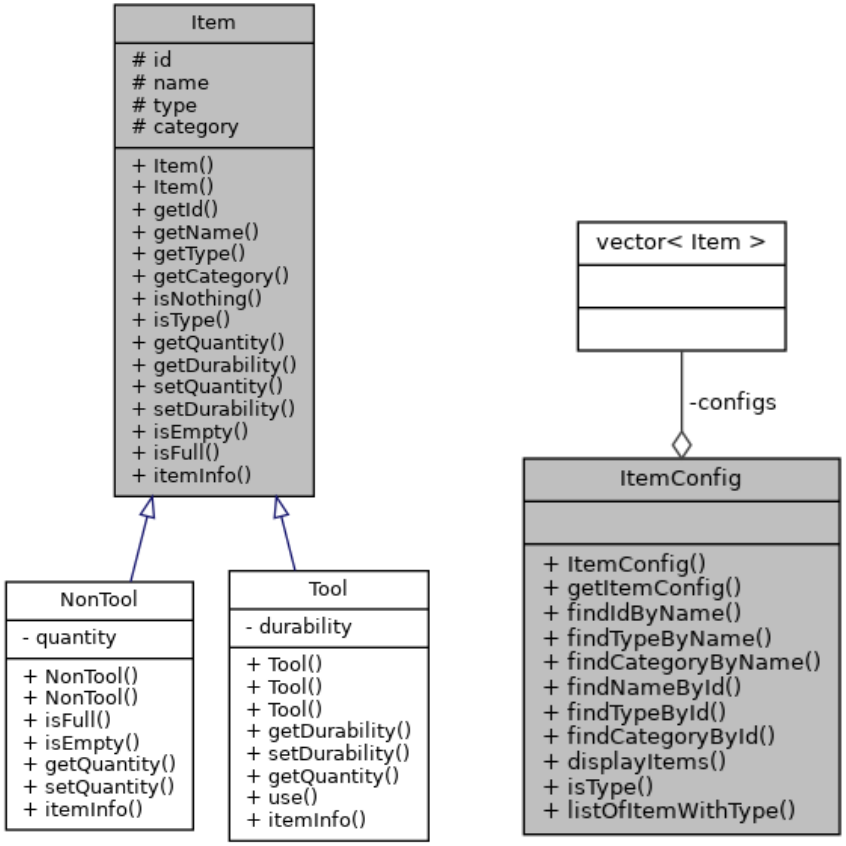


Gambar B.2 Hierarchy Diagram untuk Exception Class

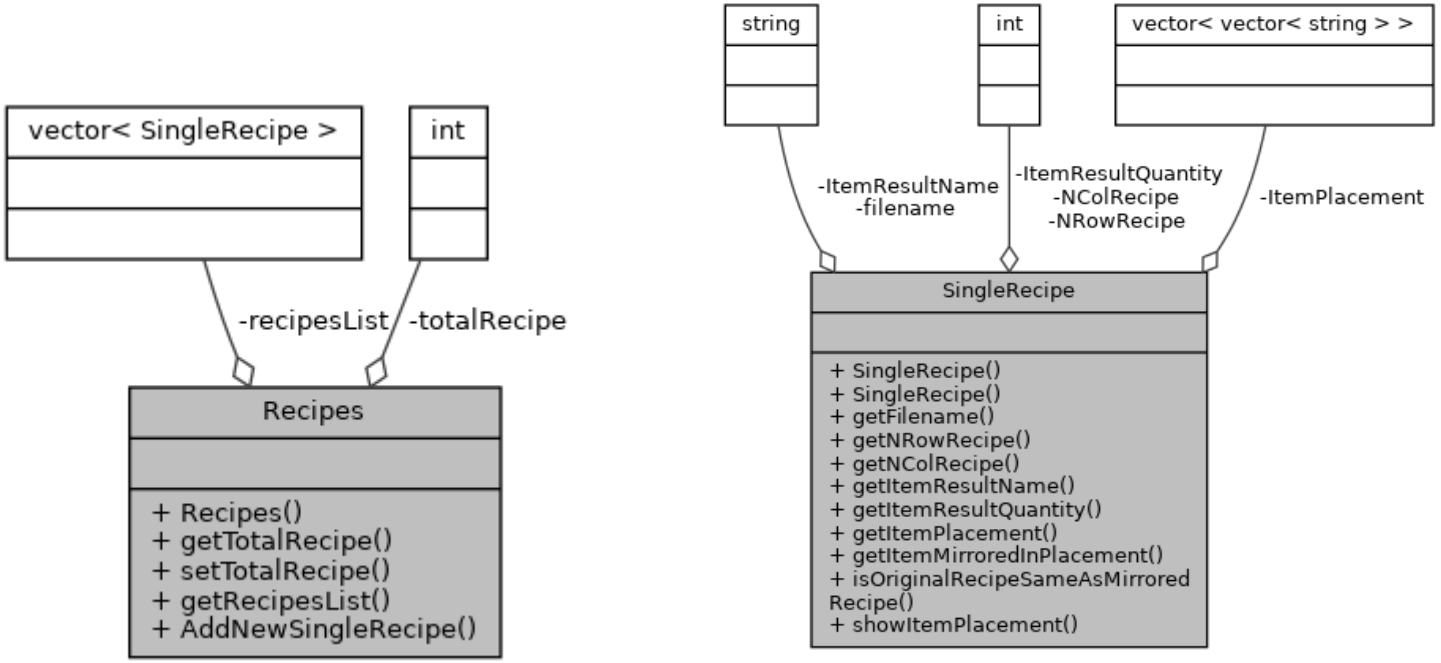
Perbesaran gambar:




BaseException
+ printMessage()
InvalidCategoryException
- ctg
+ InvalidCategoryException()
+ printMessage()
InvalidDestinationSlot
- id
+ InvalidDestinationSlot()
+ printMessage()
InvalidIDException
- id
+ InvalidIDException()
+ printMessage()
InvalidInventoryTextException
- str
+ InvalidInventoryTextException()
+ printMessage()
InvalidNameException
- itemName
+ InvalidNameException()
+ printMessage()
InvalidQuantityException
- qty
+ InvalidQuantityException()
+ printMessage()
OpenFileErrorException
- fileName
+ OpenFileErrorException()
+ printMessage()



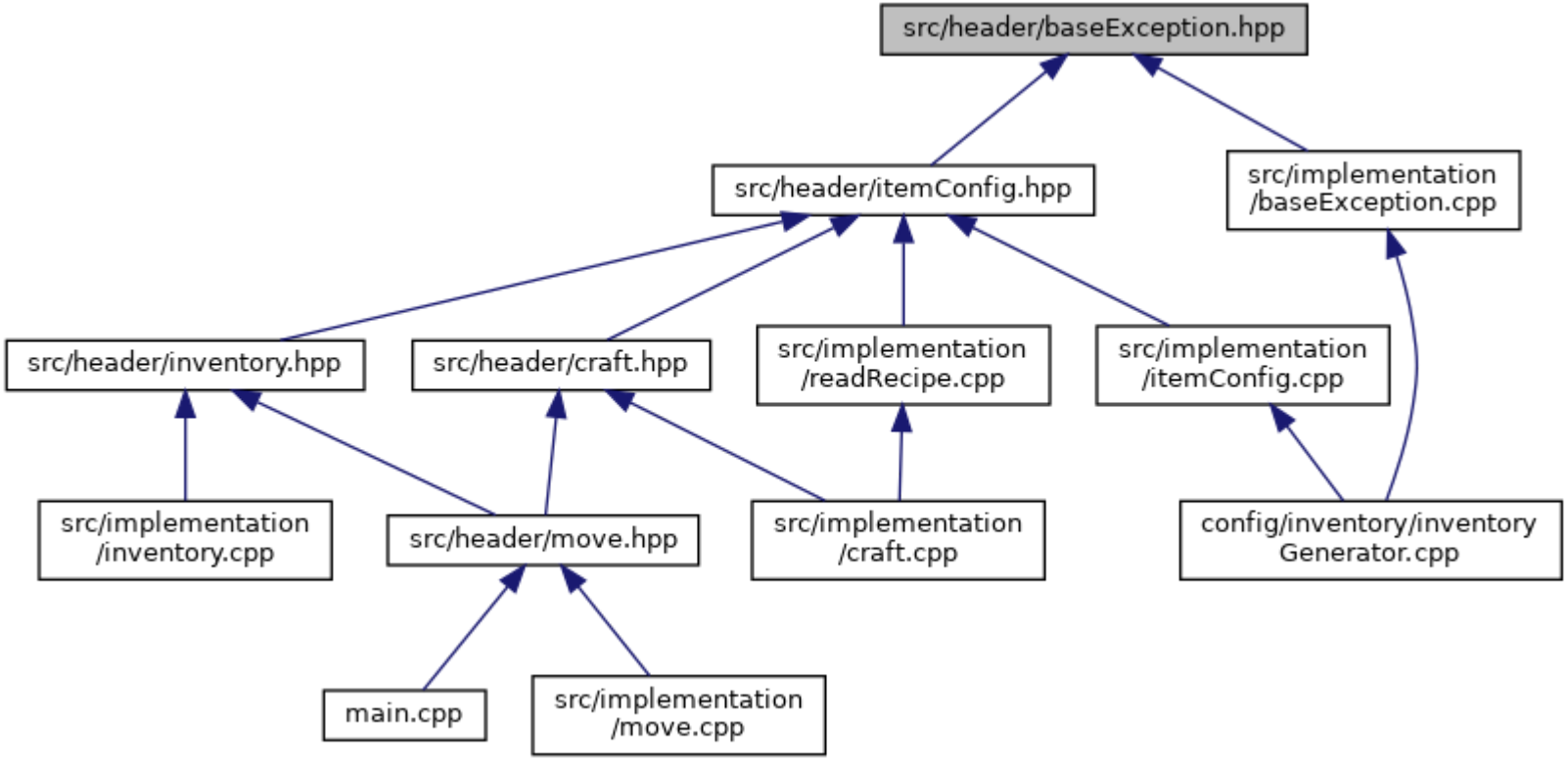
B.3 Hierarchy Diagram untuk Item dan ItemConfig



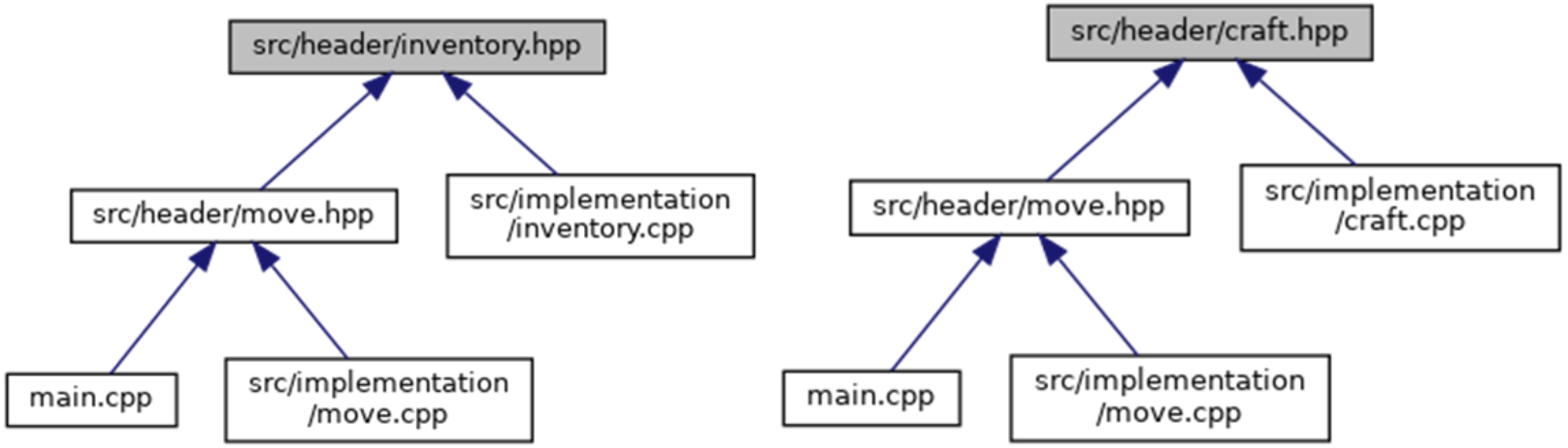
B.4 Hierarchy Diagram untuk Recipes dan SingleRecipe

C. Include Diagram untuk Masing - Masing kelas

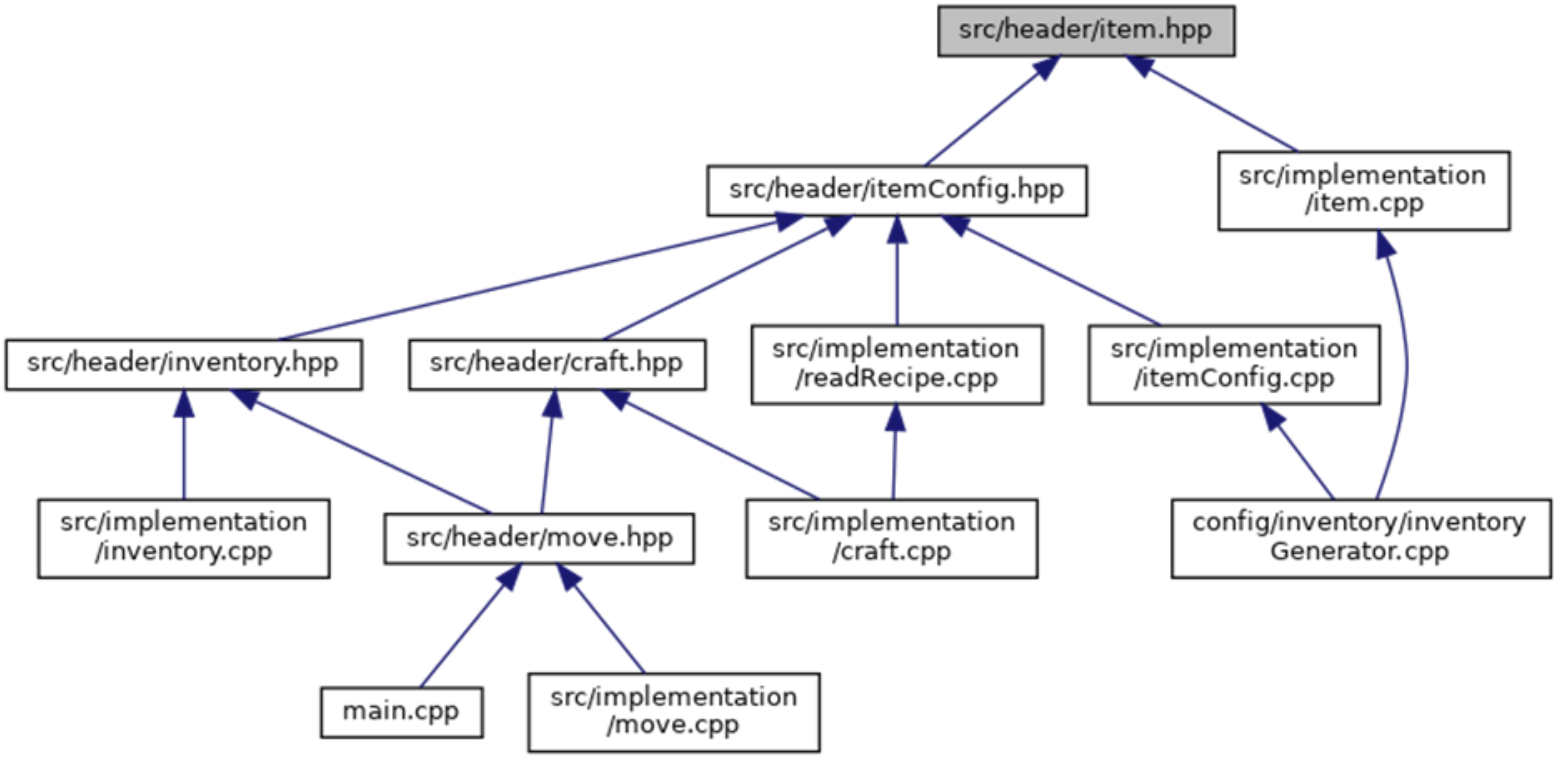
Include diagram adalah diagram yang menggambarkan kelas apa saja yang melakukan include pada class terkait.



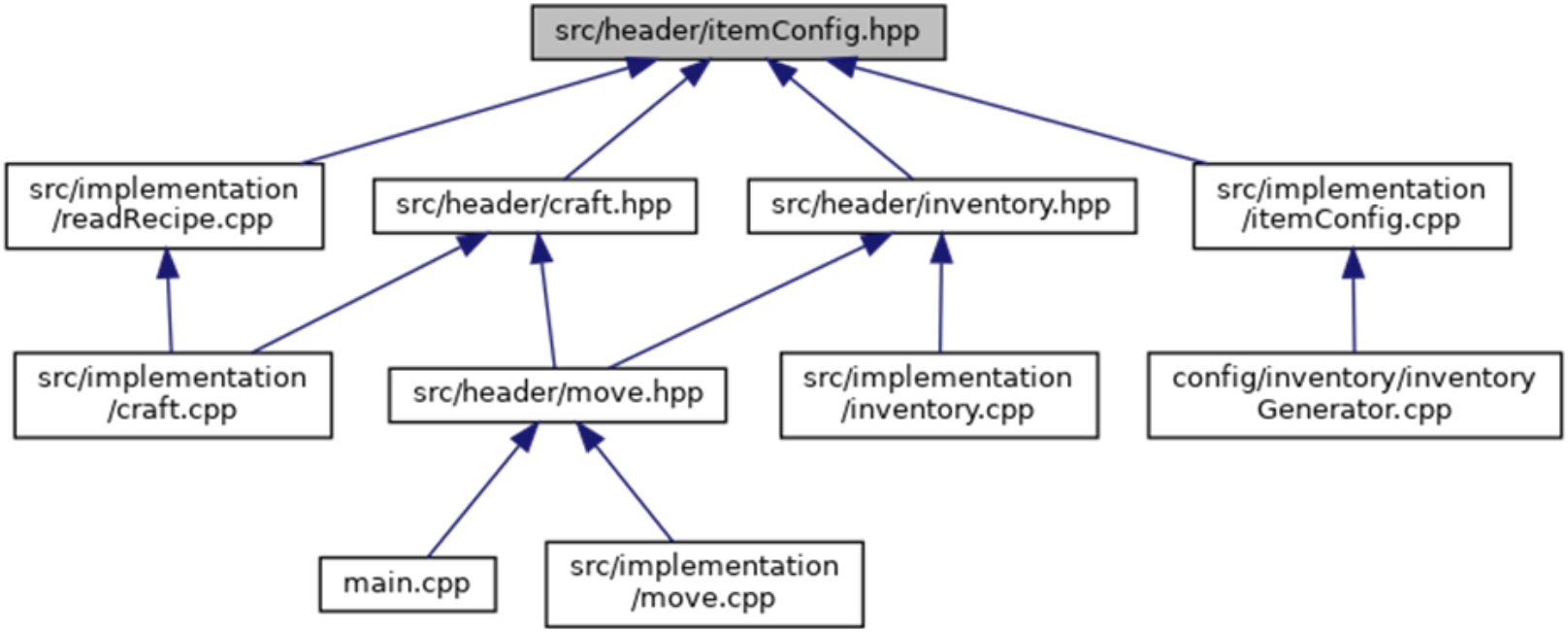
C.1 Include Diagram untuk Exception



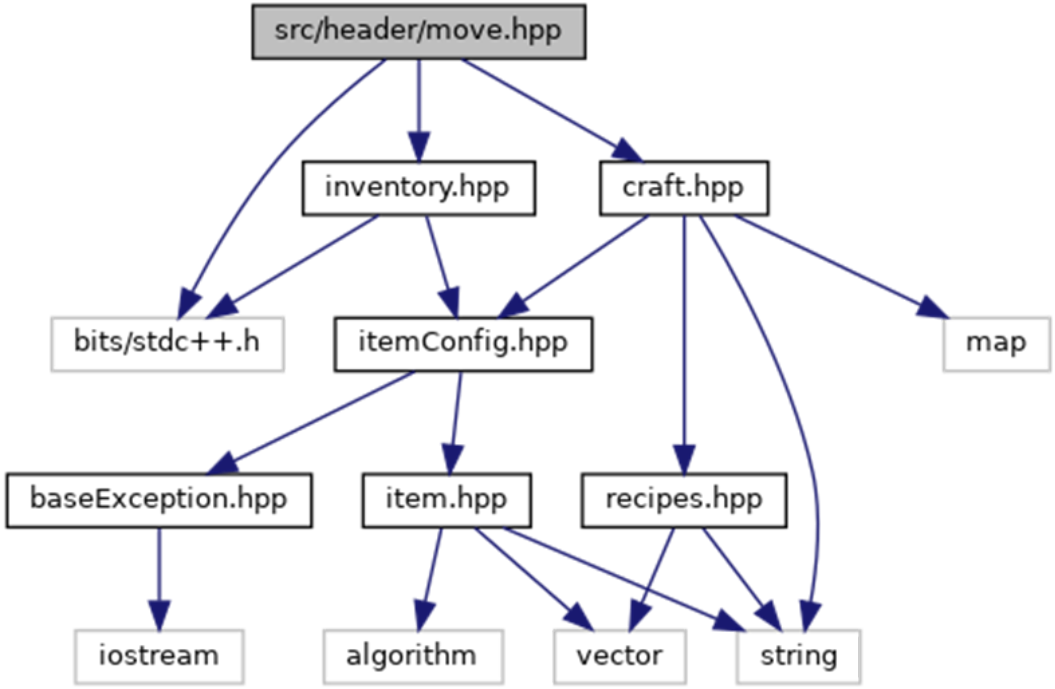
C.2 Include Diagram untuk Class Inventory dan Craft



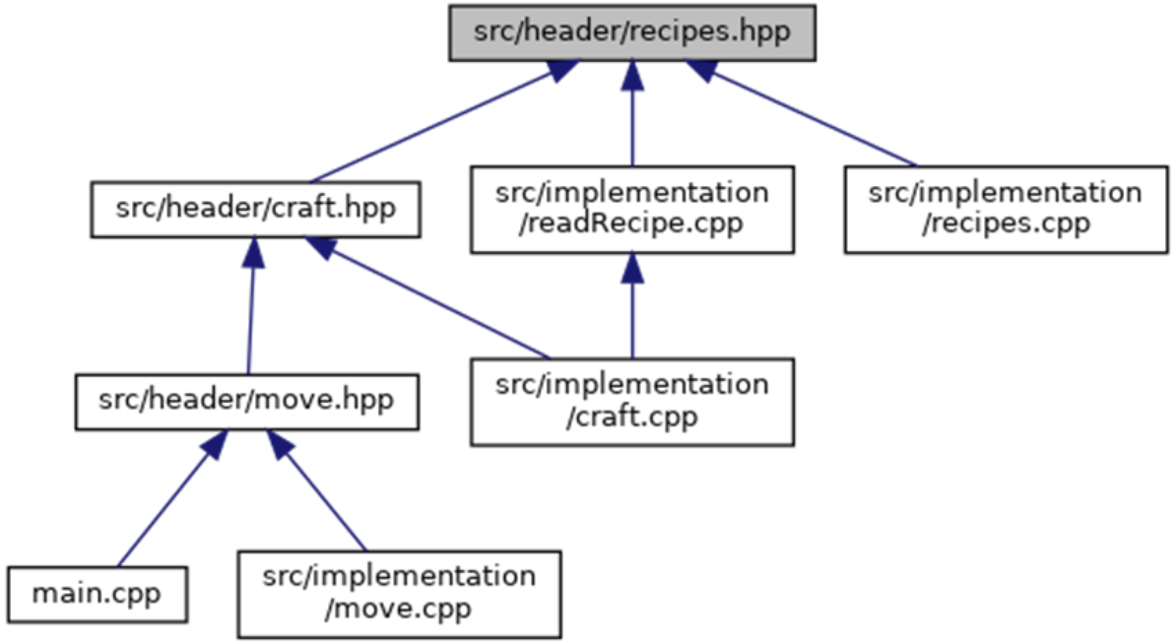
C.3 Include Diagram untuk Class Item



C.4 Include Diagram untuk Class ItemConfig



C.5 Include Diagram untuk Class Move



C.6 Include Diagram untuk Class Recipes

1.1. Recipes

Kelas recipes dibuat untuk memenuhi kebutuhan kami untuk mengetahui seberapa banyak recipe yang ada di dalam configuration file, dan menyimpannya dalam sebuah vector Single Recipe

1.2. Single Recipe

Kelas recipes dibuat untuk memenuhi kebutuhan kami untuk mengetahui seberapa banyak recipe yang ada di dalam configuration file, dan menyimpannya dalam sebuah vector Single Recipe

1.3. Craft

Craft dibuat untuk memenuhi command craft. Craft ini memuat crafting table yang didalamnya merupakan array of array bertipe pointer to item dan recipes bertipe Recipes untuk menampung resep yang digunakan.

1.4. Inventory

Kelas Inventory berguna untuk menyimpan item-item yang telah dikumpulkan oleh player. Kelas ini memiliki 2 atribut, yang pertama adalah slot berbentuk vector yang berisi item pointer berdimensi 3 x 9 dan slotUsed yang menyimpan banyaknya slot yang digunakan. Pada kelas ini dapat dilakukan beberapa method seperti show inventory, give item, discard item, use item, mengubah isi dari slot, membaca slot, menghitung jumlah item, serta mengexport isi dari inventory ke file txt.

1.5. Move

Kelas Move digunakan untuk memindahkan item-item yang player punya dari inventory ke inventory, inventory ke crafting-table, atau crafting-table ke inventory. Move dibuat menjadi kelas sendiri karena harus berkomunikasi dengan dua kelas lain sehingga akan lebih terstruktur apabila dijadikan kelas sendiri.

1.6. Item

Kelas item merupakan kelas yang menggambarkan suatu benda konkrit yang akan dilakukan proses pada tugas besar ini. Kelas ini merupakan Abstract Base Class dengan dua kelas turunan yaitu Tool dan Non-Tool, masing - masing merupakan child dari Item dengan atribut dan behaviour yang berbeda satu sama lain.

1.7. Item Config

Item config merupakan kelas yang berisi vector of item. Kelas ini berguna untuk membaca semua item yang ada dan menyimpannya dalam vector. Kelebihan penggunaan kelas ini adalah kita bisa melakukan akses terhadap semua item yang mungkin ada kapanpun, kelas ini seperti suatu database yang menyimpan semua informasi item yang ada pada permainan

1.8. NonTool

Non-Tool merupakan turunan dari kelas Item. Berupa item yang dapat ditumpuk dalam 1 slot (maksimal 64) dan umumnya merupakan benda benda material misalnya kayu atau batu

1.9. Tool

Tool merupakan turunan dari kelas Item. Tool merupakan jenis khusus dari Item yang memiliki durability, dan dapat digunakan. Tiap digunakan akan mengurangi durability dari Tool sebesar 1. Item kategori ini tidak bisa ditumpuk

1.10. Base Exception

Kelas Base Exception berisi kelas-kelas yang diperlukan untuk di throw ketika terjadi error. Contoh kasus yang dibentuk exceptionnya adalah ketika index out of range, category tidak valid, input tidak valid, destinasi slot tidak valid, dan lain-lain. Penggunaan exception ini mempermudah ketika pembuatan kelas lain untuk menghindar error-error yang tidak diinginkan.

2. Penerapan Konsep OOP

2.1. Inheritance & Polymorphism

Inheritance merupakan salah satu konsep pemrograman berorientasi objek yang memungkinkan kita untuk mendefinisikan suatu kelas berdasarkan sifat dan perilaku dari sebuah objek lain. Kelas yang menurunkan sifat disebut *parent class* dan kelas yang diturunkan disebut *child class*

2.1.1 Tool & NonTool

```
You, 8 hours ago | 1 author (You)
class Tool : virtual public Item
{
private:
    int durability;

public:
    Tool();
    Tool(int id, string name, int durability);
    Tool(int id, string name);

    int getDurability() const;
    void setDurability(int durability);
    int getQuantity() const; // bakal selalu return 1
    void use();

    void itemInfo() const;
};
```

```
class NonTool : virtual public Item
{
private:
    int quantity;

public:
    NonTool();
    NonTool(int id, string name, string type, int quantity);

    bool isFull() const;
    bool isEmpty() const;

    int getQuantity() const;
    void setQuantity(int qty);

    void itemInfo() const;
```

```
class Item
{
protected:
    int id;
    string name;
    string type;     // LOG, PLANK, STONE, -
    string category; // TOOL & NONTOOL
public:
    Item();
    Item(int id, string name, string type, string category);

    int getId() const;
    string getName() const;
    string getType() const;
    string getCategory() const;
    bool isNothing() const;      You, 1 second ago • Uncommitted changes
    bool isType(string name); // buat ngecek apakah name merupakan nama type yang bakal puny
```

Class tool dan nontool merupakan contoh penerapan inheritance, Kedua class tersebut merupakan *child class* dari kelas Item. Penggunaan konsep ini cocok diterapkan karena Class Tool pada esensinya merupakan Item dengan beberapa sifat spesifik seperti bisa digunakan (use), memiliki durability, serta mewarisi semua atribut yang dimiliki item, pada hal ini yaitu atribut ID, Nama, dan Tipe. Demikian juga halnya dengan Class NonTool.

Walaupun Tool dan NonTool keduanya merupakan item, terdapat perbedaan *behaviour* dari Tool dan NonTool . Beberapa contoh misalnya saat di Inventory, NonTool dapat ditumpuk dengan maksimal jumlah sebanyak 64 sementara Tool tidak, perbedaan lain misalnya Tool memiliki atribut durability dan dapat di-use sementara NonTool tidak. Perbedaan - perbedaan seperti ini membuat pemisahan kedua kategori menjadi class berbeda lebih masuk akal, namun kedua hal tersebut sama - sama merupakan item yang memiliki atribut umum seperti ID, Nama, dan Tipe. Jadi, penggunaan konsep Inheritance di sini menguntungkan karena kami tidak perlu mengulangi penulisan kode untuk hal yang serupa. Salah satu contoh dari keuntungan ini adalah implementasi ID dan getter ID hanya perlu dilakukan di *Parent Class* Item , hal ini lebih baik daripada harus menuliskan kode tersebut pada kedua *Child Class*.

2.2. Method/Operator Overloading

2.2.1 Item

```
Tool::Tool(int id, string name, int durability) : Item(id, name, "-", "TOOL")
{
    this->durability = durability;
}
```

```
Tool::Tool(int id, string name) : Item(id, name, "-", "TOOL")
{
    this->durability = 10;
}
```

Pada *subclass* item yang bernama Tool, terdapat dua jenis *constructor method* (selain *default constructor*) yang bisa menerima dua jenis *parameters*. Pada *constructor* pertama, terdapat *parameter* id, name, dan durability dan akan dibuat object tool dengan nilai-nilai sesuai pada *parameter constructor*. Pada *constructor* kedua, tidak ada *parameter durability* dan nilai *durability* sendiri akan dibuat sesuai default *durability* awal Tool, yaitu 10.

2.3. Template & Generic Classes

Pada program kami, kami tidak menggunakan *generic class* karena implementasi pada tiap *method* di semua *class* membutuhkan hal-hal spesifik tiap jenis atributnya, seperti pada *class* inventory, pada *method* give harus membaca tiap isi Item seperti *category, name,* dan *quantity.* Sehingga, jika tipe Item dibuat menjadi generic, tidak akan mungkin pembacaan tiap isi Item seperti *category, name,* dan *quantity.* Contoh selanjutnya ada di *class* Crafting yang harus membaca durability sebuah Tool untuk *method* craft. Hal ini tidak akan mungkin dilakukan jika Tool, yang merupakan subclass dari Item, dibuat menjadi *generic class.*

2.4. Exception

Pada kode kami, Exception diterapkan sebagai objek. Berikut beberapa Class yang menggunakan Exception

2.4.1 ItemConfig

```
class InvalidNameException : public BaseException {
private:
    string itemName;
public:
    InvalidNameException(string itemName);
    void printMessage();
};
```

```
class InvalidIDException : public BaseException {
private:
    int id;
public:
    InvalidIDException(int id);
    void printMessage();
};
```

```
string ItemConfig::findCategoryByName(string nameItem) const
{
    for (int i = 0; i < this->configs.size(); i++)
    {
        if (this->configs[i].getName() == nameItem)
            return this->configs[i].getCategory();
    }
    throw new InvalidNameException(nameItem);
}
```

Item Config merupakan Class yang memiliki tugas untuk membaca file konfigurasi dari item lalu menyimpannya dalam *vector of item*. Pada class ini , terdapat beberapa method yang berfungsi untuk mencari suatu Item berdasarkan salah satu atributnya. Pada cuplikan di atas contohnya bisa ditemukan kategori hanya dari melihat nama suatu item.

Exception cocok digunakan karena ada kemungkinan nama Item yang dicari tidak terdapat pada konfigurasi dan apabila hal ini terjadi tinggal melakukan throw InvalidNameException yang menandakan tidak terdapat item dengan nama tersebut. Keuntungan disini adalah fungsi findCategoryByName tidak perlu mengeluarkan keluaran apa - apa , hanya perlu melakukan throw dan pengguna akan langsung mengetahui bahwa nama yang diberikan tidak valid.

2.4.2 Inventory

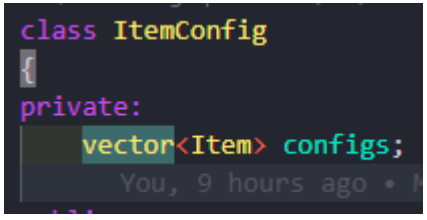
```
void Inventory::discard(string slotId, int itemQty)
{
    if (itemQty == 0) return;

    slotId.erase(0,1);
    int idNum;

    try {
        idNum = stoi(slotId);
    } catch(exception &err) {
        throw new CustomException("stoi error");
    }

    int row = idNum/COLSLOT;
    int col = idNum%COLSLOT;

    if (row < 0 || row >= ROWSLOT) {
        throw new IndexOutOfRangeException(row);
    }

    if (col < 0 || col >= COLSLOT) {
        throw new IndexOutOfRangeException(col);
    }

    if (slot[row][col]->isNothing()){
        throw new EmptySlotException(row, col);
    }

    if (itemQty < 0){
        throw new InvalidQuantityException(itemQty);
    }
```

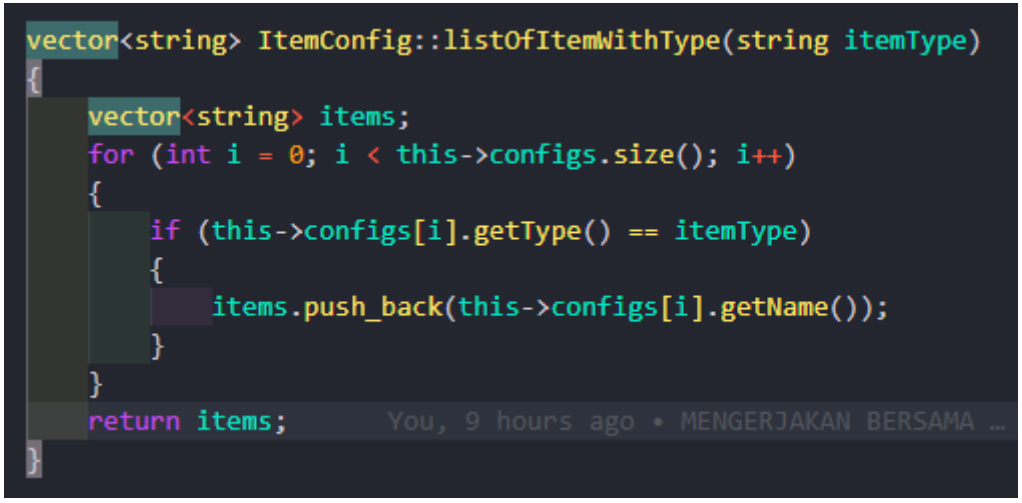
```
try {
    id = stoi(idItem);
    qty = stoi(qtyItem);
} catch(exception &err) {
    throw new CustomException("stoi error");
}

if (id < 1 || id > 38) {
    throw new IndexOutOfRangeException(id);
}

string ctg = readItemConfig.findCategoryById(id);
if (ctg == "NONTOOL" && (qty < 1 || qty > MAXQTY)
) {
    throw new InvalidQuantityException(qty);
}

if (ctg != "NONTOOL" && ctg != "TOOL"){
    throw new CustomException(ctg);
}

string itemName = readItemConfig.findNameById(id);
```

Inventory merupakan class yang memiliki tugas memanajemen data item dengan vector 2 dimensi berukuran 3x9. Pada class ini banyak sekali digunakan exception karena ada banyak kemungkinan error pada input, seperti index di luar batas, slot inventory kosong, quantity invalid (tidak memenuhi kriteria), error pada stoi, dan lain-lain. Dengan adanya class Base Exception, semua error tersebut dapat diatasi dengan sempurna.

2.5. C++ Standard Template Library

2.5.1 Vector





Beberapa kelas pada kode kami mengimplementasikan STL vector, seperti pada class crafting, itemconfig, dan recipe. STL ini digunakan karena vector merupakan array dinamis yang penyimpanannya sudah ditangani secara otomatis. Untuk contoh misal dapat dilihat kedua cuplikan kode di atas. Fungsi listOfItemWithType akan mengembalikan sekumpulan

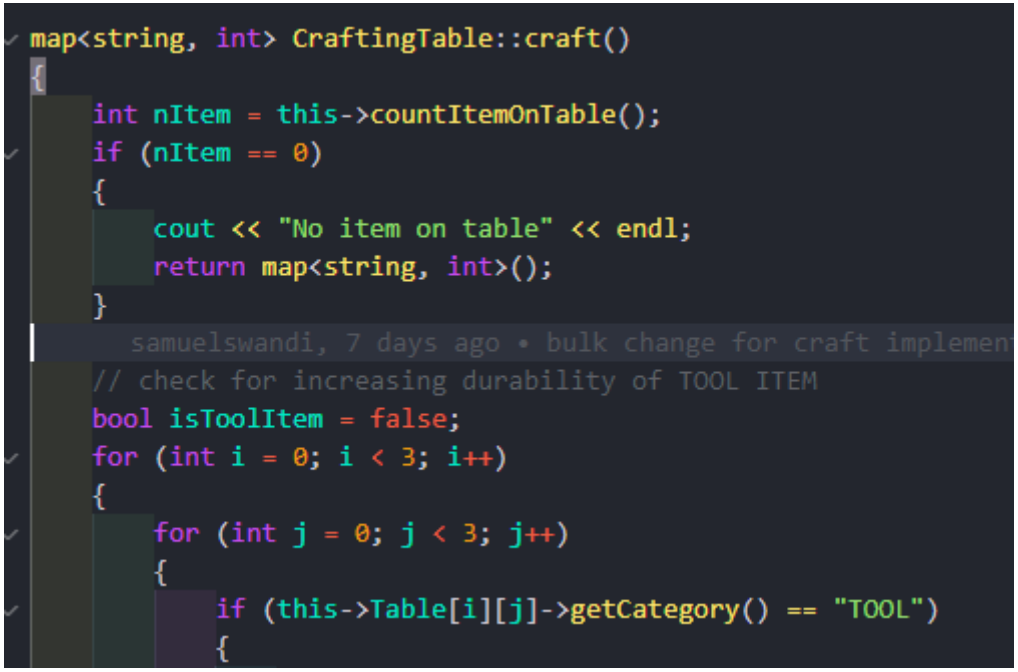
item dengan tipe tertentu, penggunaan vector di sini akan lebih mudah ketimbang penggunaan list/array karena jumlah elemen yang akan dikembalikan tidak diketahui sebelumnya, sehingga kita tidak tahu berapa besar array yang perlu dialokasikan. Sementara dengan penggunaan vector, kami hanya perlu memasukkan string dan size nya akan berubah secara otomatis.

2.5.2 Fstream

```
void Inventory::exportFile(string fileName){
    ofstream fout;
    string filePath = fileName;
    fout.open(filePath);
    for(int i=0; i<27; i++){
        if(slotItem(i)->isNothing()){
            fout << "0:0" << endl;
        }
        else if(slotItem(i)->getCategory() == "TOOL"){
            fout << slotItem(i)->getId() << ":" << slotItem(i)->getDurability() << endl;
        } else {
            fout << slotItem(i)->getId() << ":" << slotItem(i)->getQuantity() << endl;
        }
    }
    fout.close();
}
```

Ketika menuliskan file pada suatu file txt, diperlukan sebuah library fstream pada C++. Library ini berguna untuk membaca file dari suatu text (ifstream) serta menuliskan file ke suatu text (ofstream). Salah satu penggunaan library ini berada pada method exportFile dan importFile pada class Inventory. Library ini digunakan untuk menuliskan isi dari inventory saat itu ke file txt yang sudah ditargetkan pathnya.

2.5.3 Map



```
map<string, int> CraftingTable::craft()
{
    int nItem = this->countItemOnTable();
    if (nItem == 0)
    {
        cout << "No item on table" << endl;
        return map<string, int>();
    }
      samuelswandi, 7 days ago • bulk change for craft implemen
    // check for increasing durability of TOOL ITEM
    bool isToolItem = false;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (this->Table[i][j]->getCategory() == "TOOL")
            {
```

Map merupakan STL berupa kontainer yang menyimpan elemen yang memiliki key value dan mapped value. STL ini digunakan pada craft untuk mengembalikan nilai apabila craft berhasil dilakukan berupa nama item dan jumlah dari hasil crafting. Apabila craft gagal dilakukan map akan mengembalikan nilai <-,0> dan program akan mengetahui berarti crafting gagal dilakukan.

2.6. Virtual Function

```
virtual int getQuantity() const;
virtual int getDurability() const;
virtual void setQuantity(int qty);
virtual void setDurability(int durability);
virtual bool isEmpty() const;
virtual bool isFull() const;
```

Kelas item memiliki beberapa virtual function seperti yang tertera diatas. Konsep ini digunakan karena sesuai dengan spesifikasi dimana misalnya kelas item sendiri itu tidak memiliki quantity dan durability, sehingga untuk implementasi dilimpahkan pada anaknya, misalnya setQuantity dilimpahkan pada class nonTool karena hanya nonTool yang bisa ditumpuk dan setDurability dan getDurability dilimpahkan pada class Tool.

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Multiple Crafting

- Sebelum

```
INPUT COMMAND: SHOW
Crafting Table <ItemName Quantity/Durability>:
[ [C0: OAK_LOG 2 ] [C1: EMPTY ] [C2: EMPTY ] ]
[ [C3: EMPTY ] [C4: EMPTY ] [C5: EMPTY ] ]
[ [C6: EMPTY ] [C7: EMPTY ] [C8: EMPTY ] ]
Inventory:
[I00: EMPTY] [I01: EMPTY] [I02: EMPTY] [I03: EMPTY] [I04: EMPTY] [I05: EMPTY] [I06: EMPTY] [I07: EMPTY] [I08: EMPTY]
[I09: EMPTY] [I10: EMPTY] [I11: EMPTY] [I12: EMPTY] [I13: EMPTY] [I14: EMPTY] [I15: EMPTY] [I16: EMPTY] [I17: EMPTY]
[I18: EMPTY] [I19: EMPTY] [I20: EMPTY] [I21: EMPTY] [I22: EMPTY] [I23: EMPTY] [I24: EMPTY] [I25: EMPTY] [I26: EMPTY]
```

- Craft Pertama

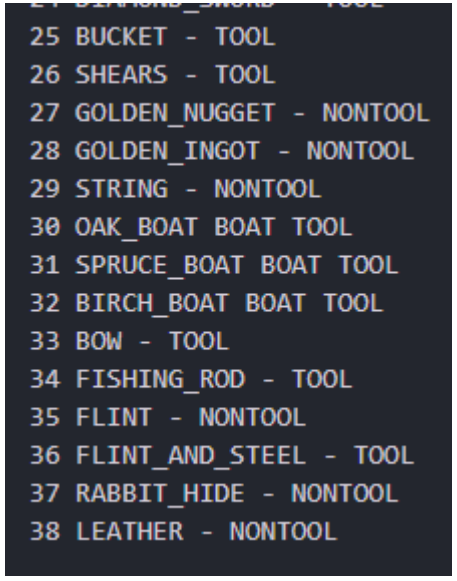
```
INPUT COMMAND: CRAFT
Crafting success
Created : 4 OAK_PLANK

INPUT COMMAND: SHOW
Crafting Table <ItemName Quantity/Durability>:
[ [C0: OAK_LOG 1 ] [C1: EMPTY ] [C2: EMPTY ] ]
[ [C3: EMPTY ] [C4: EMPTY ] [C5: EMPTY ] ]
[ [C6: EMPTY ] [C7: EMPTY ] [C8: EMPTY ] ]
Inventory:
[I00: OAK_PLANK 4] [I01: EMPTY] [I02: EMPTY] [I03: EMPTY] [I04: EMPTY] [I05: EMPTY] [I06: EMPTY] [I07: EMPTY] [I08: EMPTY]
[I09: EMPTY] [I10: EMPTY] [I11: EMPTY] [I12: EMPTY] [I13: EMPTY] [I14: EMPTY] [I15: EMPTY] [I16: EMPTY] [I17: EMPTY]
[I18: EMPTY] [I19: EMPTY] [I20: EMPTY] [I21: EMPTY] [I22: EMPTY] [I23: EMPTY] [I24: EMPTY] [I25: EMPTY] [I26: EMPTY]
```

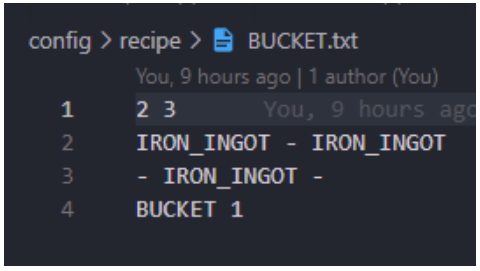
- Craft Kedua

```
INPUT COMMAND: CRAFT
Crafting success
Created : 4 OAK_PLANK

INPUT COMMAND: SHOW
Crafting Table <ItemName Quantity/Durability>:
[ [C0: EMPTY ] [C1: EMPTY ] [C2: EMPTY ] ]
[ [C3: EMPTY ] [C4: EMPTY ] [C5: EMPTY ] ]
[ [C6: EMPTY ] [C7: EMPTY ] [C8: EMPTY ] ]
Inventory:
[I00: OAK_PLANK 8] [I01: EMPTY] [I02: EMPTY] [I03: EMPTY] [I04: EMPTY] [I05: EMPTY] [I06: EMPTY] [I07: EMPTY] [I08: EMPTY]
[I09: EMPTY] [I10: EMPTY] [I11: EMPTY] [I12: EMPTY] [I13: EMPTY] [I14: EMPTY] [I15: EMPTY] [I16: EMPTY] [I17: EMPTY]
[I18: EMPTY] [I19: EMPTY] [I20: EMPTY] [I21: EMPTY] [I22: EMPTY] [I23: EMPTY] [I24: EMPTY] [I25: EMPTY] [I26: EMPTY]

INPUT COMMAND:
```

Untuk bonus ini, kami mengimplementasikannya dengan penggunaan CRAFT berkali-kali sampai tidak ada lagi barang ataupun recipe yang memenuhi. Dapat dilihat pada craft OAK_LOG di atas. Untuk 1 OAK_LOG yang di-CRAFT maka akan terbentuk 4 OAK_PLANK, dan karena hanya ada 1 recipe yang dapat menandakan bahwa hanya ada 1 OAK_LOG yang akan menghasilkan 4 OAK_PLANK, maka pada CRAFT pertama, terdapat sisa 1 OAK_LOG yang nantinya dapat di-CRAFT lagi.

3.1.2. Item dan Tool Baru





Untuk bonus ini, hanya perlu menambahkan item pada file konfigurasi pada item.txt dan apabila item tersebut dapat dibuat dari item lain ditambahkan juga resepnya dengan referensi dari https://www.minecraft-crafting.net/ .

3.2. Bonus Kreasi Mandiri

3.2.1. ASCII Banner Pijat C++

Pembuatan ASCII Banner ini digunakan ketika dilakukan run pertama kali pada program. Banner ini menampilkan nama kelompok kami yaitu Pijac C++ dan tulisan “MINECRAFT CLONE V.01” yang berarti versi pertama dari pembuatan program kloning dari Minecraft.

3.2.2. Command HELP

```
INPUT COMMAND: HELP
1. SHOW
2. GIVE <ITEM_NAME> <ITEM_QTY>
3. DISCARD <INVENTORY_SLOT_ID> <ITEM_QTY>
4. MOVE <INVENTORY_SLOT_ID> N <CRAFTING_SLOT_ID_1>
5. MOVE <INVENTORY_SLOT_ID_SRC> 1 <INVENTORY_SLOT_ID_DEST>
6. MOVE <CRAFTING_SLOT_ID> 1 <INVENTORY_SLOT_ID>
7. USE <INVENTORY_SLOT_ID>
8. CRAFT
9. EXPORT <NAMA_FILE>
```

Pembuatan command HELP berguna untuk player yang pertama kali baru menjalankan program ini. Command ini akan menampilkan seluruh command valid untuk menjalankan program. Dengan adanya command ini, maka semua orang dapat memainkan game ini tanpa kebingungan.

3.2.3. Inventory Generator

```cpp
#include <bits/stdc++.h>
#include "../../src/implementation/baseException.cpp"
#include "../../src/implementation/itemConfig.cpp"
#include "../../src/implementation/item.cpp"

using namespace std;

int main() {
    srand((unsigned)time(0));

    string configPath = "..";
    string fileName = "item.txt";
    ItemConfig readItemConfig = ItemConfig(configPath, fileName);

    ofstream inv;
    inv.open("inventory.txt");

    for(int i=0; i<27; i++){
        int cur = rand();
        if(cur%4 == 0) inv << "0:0" << '\n';
        else {
            int idItem = rand()%38 + 1;
            if(readItemConfig.findCategoryById(idItem) == "TOOL"){
                inv << idItem << ":" << (rand() % 10 + 1) << '\n';
            } else {
                inv << idItem << ":" << (rand() % 64 + 1) << '\n';
            }
        }
    }

    inv.close();
}
```

Inventory generator merupakan sebuah program yang akan membuat file txt baru berisi 27 line yang menyatakan slot inventory dengan format sama seperti ketika mengexport file (format id:quantity untuk non tool dan id:durability untuk tool). Tujuan dari pembentukan program ini adalah mempermudah tester ketika hendak mengisi inventorynya terlebih dahulu sehingga tester bisa melakukan testing terhadap kasus-kasus yang lebih *advance*.

4. Pembagian Tugas

Modul (dalam poin spek)	Implementer	Tester
Exception	13520036	13520036
Item	13520048, 13520081	13520048, 13520081
ItemConfig	13520048	13520036, 13520048
Inventory	13520036, 13520132	13520036, 13520081
Move	13520081, 13520132	13520036, 13520048, 13520081, 13520132
Craft	13520075, 13520078	13520081, 13520075, 13520078
Recipes	13520075, 13520078	13520075, 13520078
Main	13520036, 13520048, 13520075, 13520081	13520036, 13520048, 13520081, 13520075, 13520078