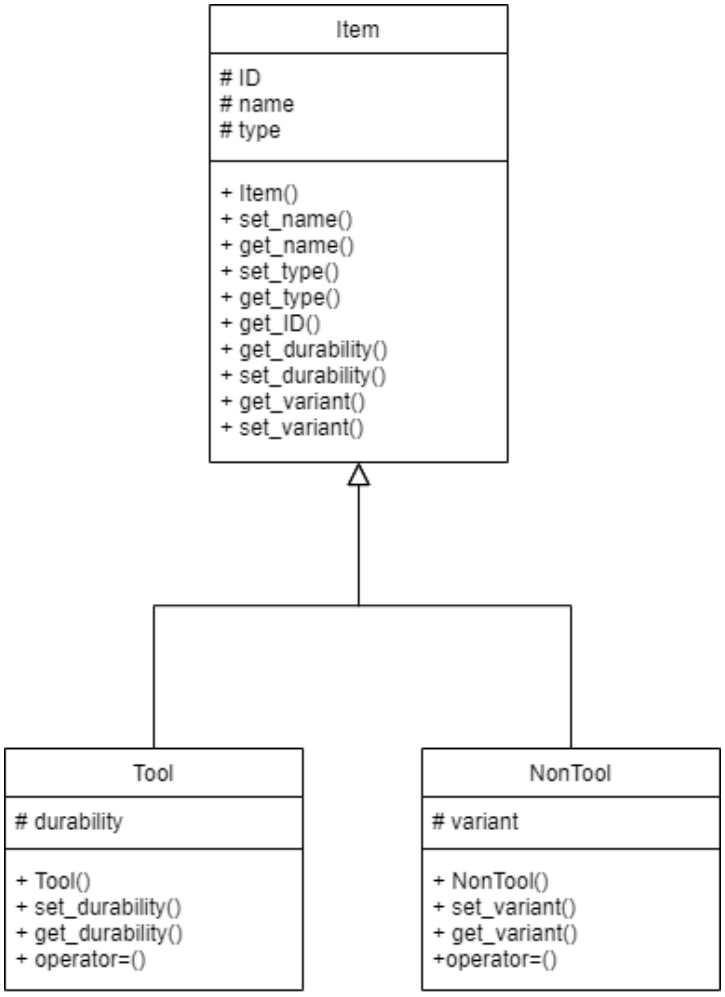
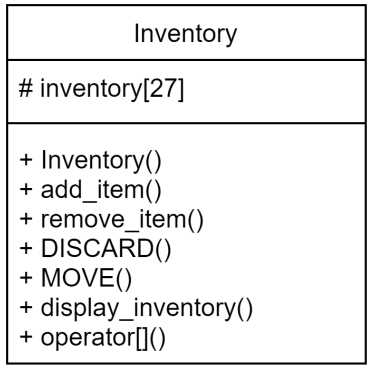
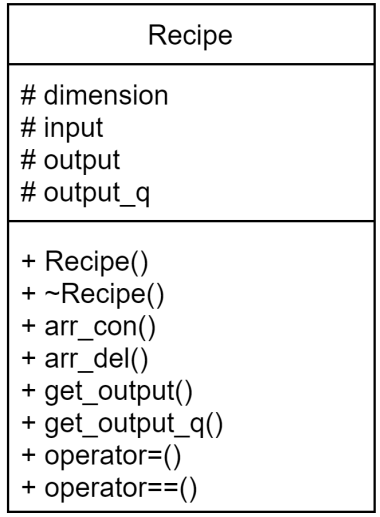
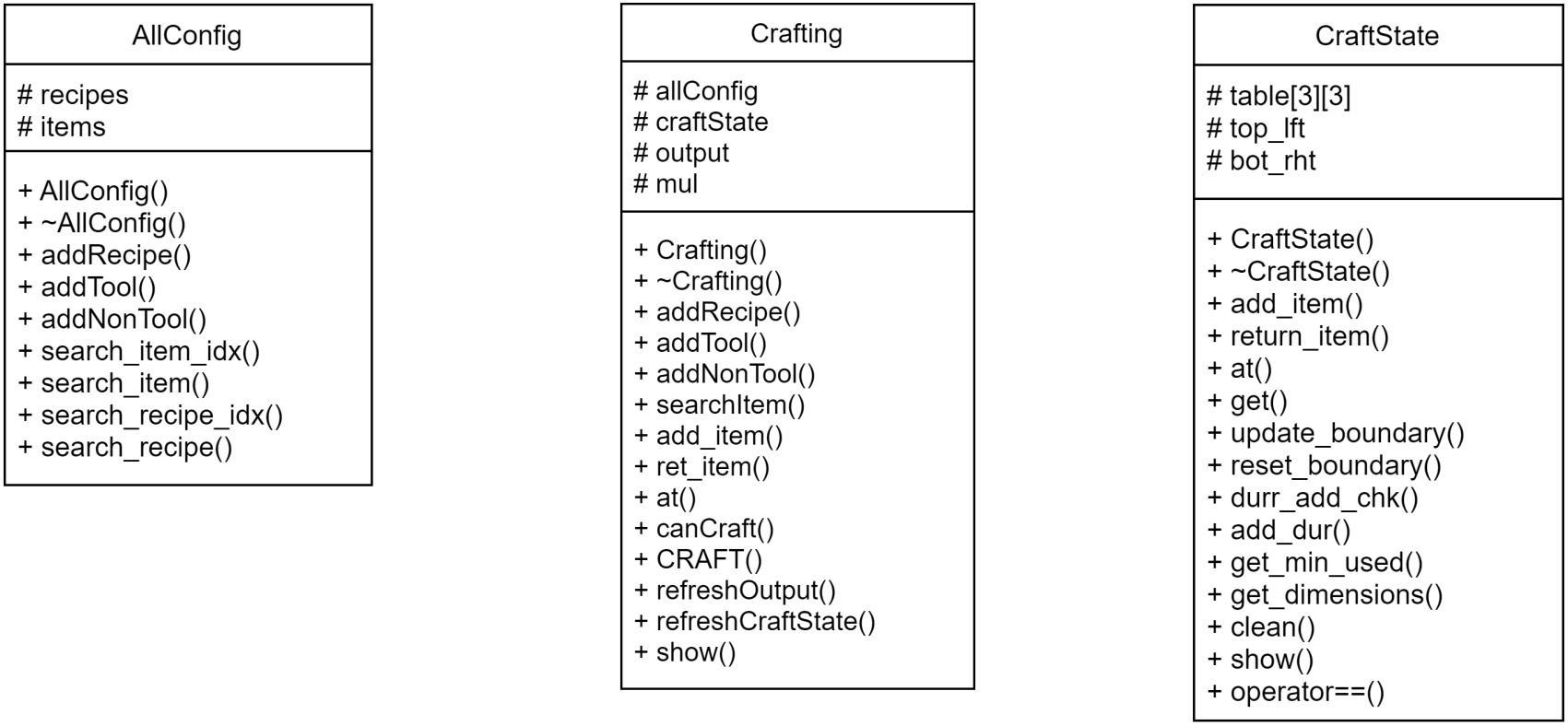
Kelas : 03
Nomor Kelompok : 03
Nama Kelompok : OOPs
1. 13520006 / Vionie Novencia Thanggestyo
2. 13520015 / Jaya Mangalo Soegeng Rahardjo
3. 13520093 / Vincent Ho
4. 13520135 / Muhammad Alif Putra Yasa
5. 13520165 / Ghazian Tsabit Alkamil
Asisten Pembimbing : Fariz Rizki Ekananda

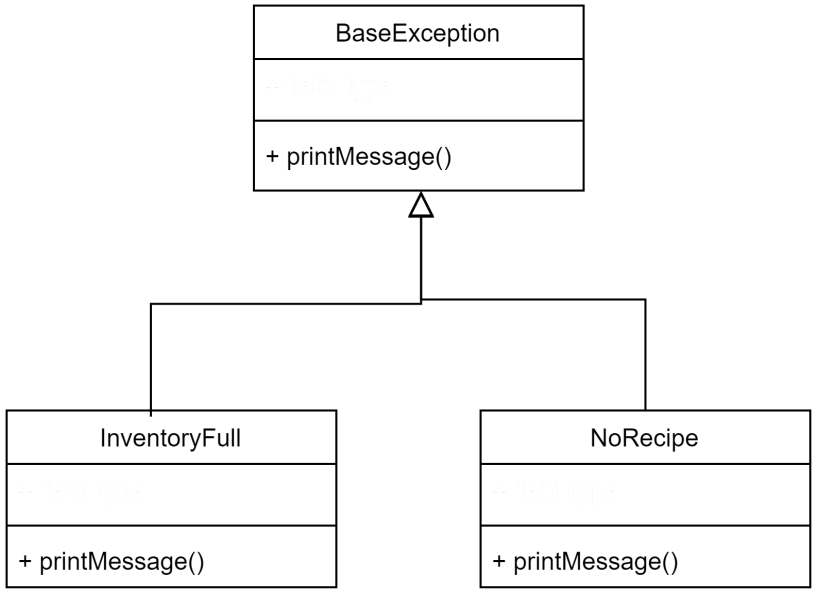
1. Diagram Kelas

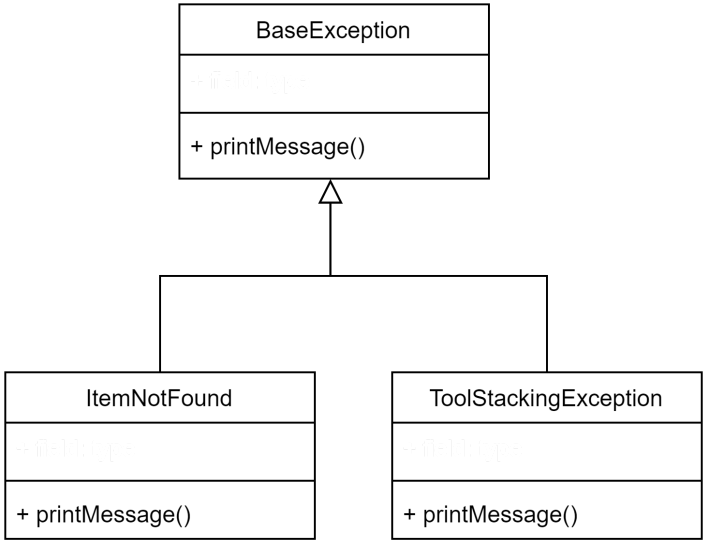


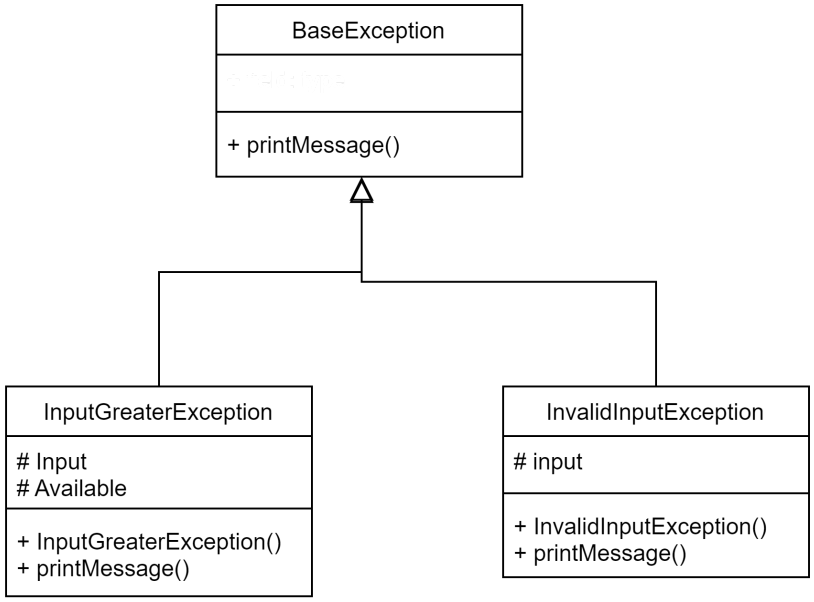


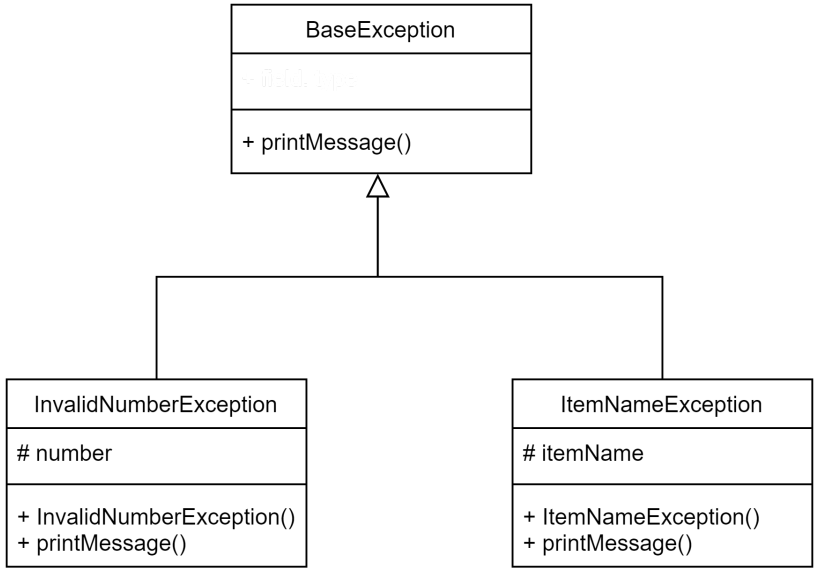


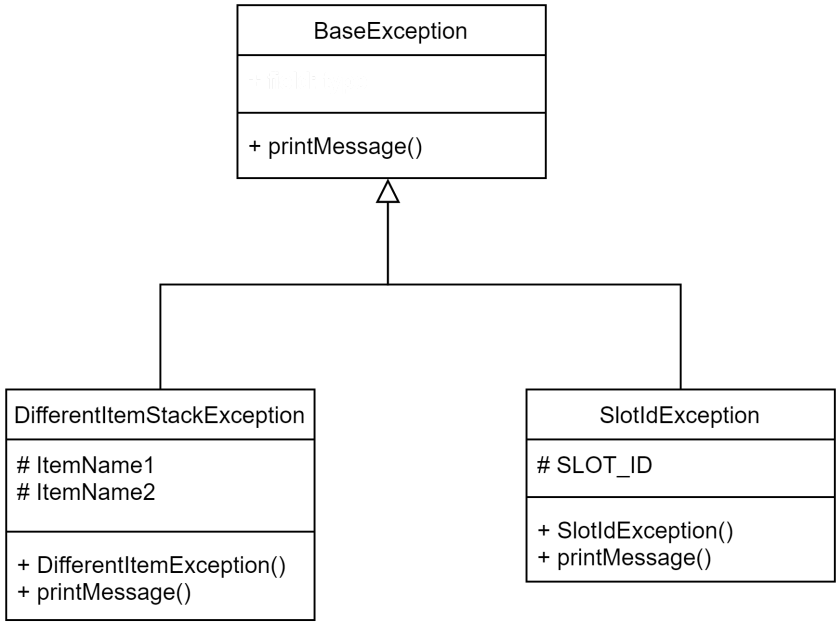

AllConfig
recipes
items
+ AllConfig()
+ ~AllConfig()
+ addRecipe()
+ addTool()
+ addNonTool()
+ search_item_idx()
+ search_item()
+ search_recipe_idx()
+ search_recipe()
Crafting
allConfig
craftState
output
mul
+ Crafting()
+ ~Crafting()
+ addRecipe()
+ addTool()
+ addNonTool()
+ searchItem()
+ add_item()
+ ret_item()
+ at()
+ canCraft()
+ CRAFT()
+ refreshOutput()
+ refreshCraftState()
+ show()
CraftState
table[3][3]
top_lft
bot_rht
+ CraftState()
+ ~CraftState()
+ add_item()
+ return_item()
+ at()
+ get()
+ update_boundary()
+ reset_boundary()
+ durr_add_chk()
+ add_dur()
+ get_min_used()
+ get_dimensions()
+ clean()
+ show()
+ operator==()

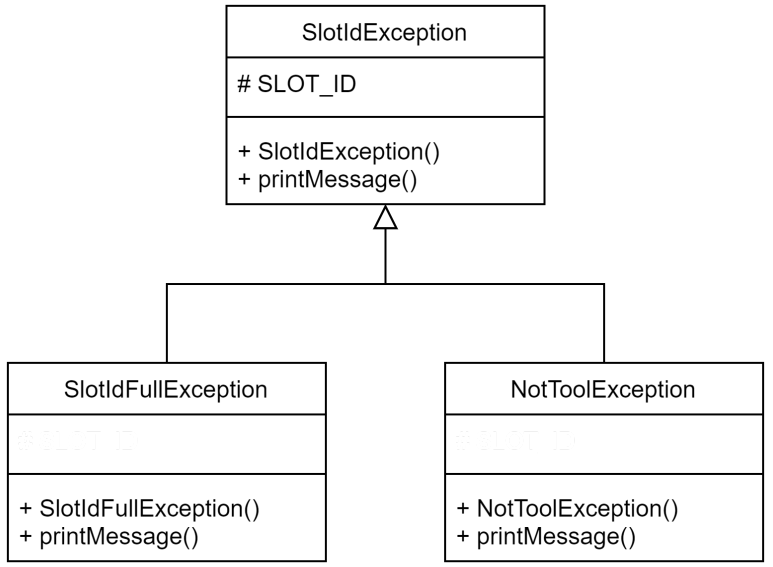

BaseException
+ printMessage()
InventoryFull
+ printMessage()
NoRecipe
+ printMessage()

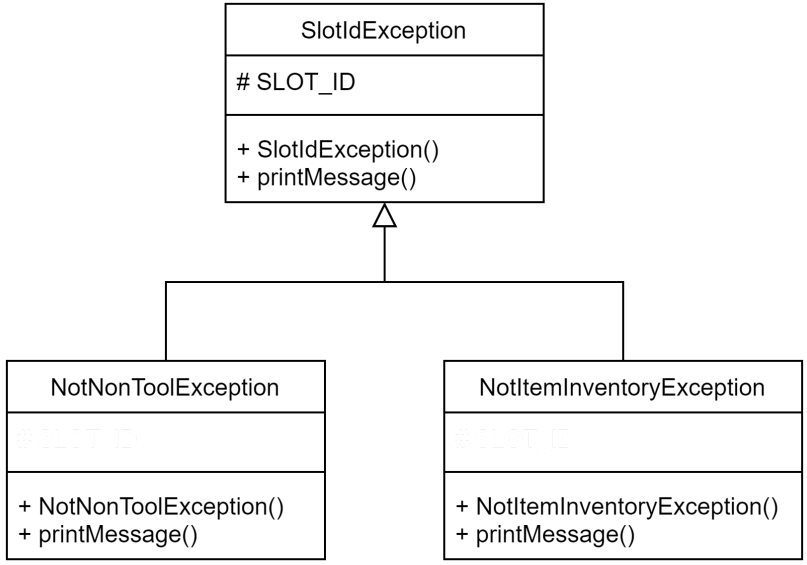

BaseException
+ printMessage()
ItemNotFound
+ printMessage()
ToolStackingException
+ printMessage()

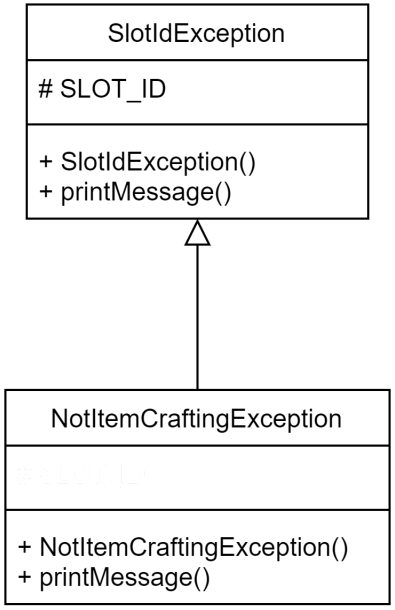

BaseException
+ printMessage()
InputGreaterException
Input
Available
+ InputGreaterException()
+ printMessage()
InvalidInputException
input
+ InvalidInputException()
+ printMessage()


BaseException
+ printMessage()
InvalidNumberException
number
+ InvalidNumberException()
+ printMessage()
ItemNameException
itemName
+ ItemNameException()
+ printMessage()


BaseException
+ printMessage()
DifferentItemStackException
ItemName1
ItemName2
+ DifferentItemException()
+ printMessage()
SlotIdException
SLOT_ID
+ SlotIdException()
+ printMessage()


SlotIdException
SLOT_ID
+ SlotIdException()
+ printMessage()
SlotIdFullException
+ SlotIdFullException()
+ printMessage()
NotToolException
+ NotToolException()
+ printMessage()


SlotIdException
SLOT_ID
+ SlotIdException()
+ printMessage()
NotNonToolException
+ NotNonToolException()
+ printMessage()
NotItemInventoryException
+ NotItemInventoryException()
+ printMessage()


SlotIdException
SLOT_ID
+ SlotIdException()
+ printMessage()
NotItemCraftingException
+ NotItemCraftingException()
+ printMessage()

2. Penerapan Konsep OOP

2.1 Inheritance & Polymorphism

Konsep Inheritance digunakan pada kelas item. Kelas item memiliki atribut id, name, dan type. Jika type item adalah tool, maka akan terdapat tambahan atribut durability dan tambahan atribut variant untuk nontool. Oleh karena itu, kelas item cocok menggunakan konsep inheritance dengan kelas item sebagai parent, dan kelas tool dan nontool sebagai child. Kelebihannya adalah code menjadi lebih ringkas dan efisien, kelas item hanya perlu dideklarasikan sekali. Untuk item dengan tipe berbeda, hanya perlu dideklarasikan atribut spesialnya saja seperti atribut durability pada tool dan variant pada nontool. Masing-masing kelas child tidak perlu mendeklarasikan ulang atribut yang sudah ada pada kelas parent.

Konsep Inheritance dan polymorphism juga dipakai di exception handling, Kelas BaseException memiliki banyak anak dan kelas BaseException di construct/polymorph oleh anaknya sehingga fungsi printMessage BaseException akan di-overwrite oleh fungsi anaknya. Untuk lebih lanjut di 2.4 Exception.

```
class Item {
    private:
        int ID;
        std :: string name;
        std :: string type;
    public:
        Item();
        Item(int ID, std :: string name, std :: string type);
        void set_name(std :: string name);
        std :: string get_name() const;
        void set_type(std :: string type);
        std :: string get_type() const;
        int get_ID() const;
        virtual int get_durability() const;
        virtual void set_durability(int durability);
        virtual std :: string get_variant() const;
        virtual void set_variant(std :: string variant);
};

class Tool : public Item{
    private:
        int durability;
    public:
        Tool();
        Tool(int ID, std :: string name);
        void set_durability(int durability);
        int get_durability() const;
        void operator=(const Tool &t);
};

class NonTool : public Item{
    private:
        std :: string variant;
    public:
        NonTool();
        NonTool(int ID, std :: string name, std :: string variant);
        void set_variant(std :: string variant);
        std :: string get_variant() const;
        void operator=(const NonTool &nt);
};
#endif
```

2.2 Method/Operator Overloading

```
void add_item(int inventoryID, Item *item, int quantity);
void add_item(Item *item, int quantity);

void add_item(Tool item, int quantity);
void add_item(NonTool item, int quantity);

void add_item(int inventoryID, Tool item, int quantity);
void add_item(int inventoryID, NonTool item, int quantity);
```

Fungsi add_item(int inventoryID, Item *item, int quantity) digunakan pada command *MOVE* khususnya apabila memindahkan item ke inventory dengan lokasi yang spesifik. Fungsi add_item(Item*, int) digunakan pada command CRAFT ketika menambahkan hasil Crafting ke Inventory. Selain itu, fungsi add_item(Item*, int) juga digunakan pada command GIVE, ketika kita ingin menambahkan item baru ke inventory.

Fungsi add_item() yang kelompok kami buat menggunakan konsep dynamic_cast pada C++, agar ketika kita menambahkan objek yang memiliki kelas Item ke dalam inventory, program akan langsung mengetahui tipe item tersebut apakah dia item dengan tipe Tool atau NonTool. Penggunaan konsep ini sesuai dengan kasus diatas, karena ketika kita ingin menambahkan item ke dalam inventory, item tersebut langsung bisa mengakses method dari child nya.

```
friend bool operator==(const CraftState &cs, const Recipe &r);
friend bool operator==(const Recipe &r, const CraftState &cs);
```

Operator == diatas digunakan untuk menentukan kesamaan antara tabel Crafting dengan resep. Alasan digunakannya operator == adalah untuk memanfaatkan STL Algorithm find, yang akan mencari Resep yang cocok dengan tabel Crafting.

2.3 Template & Generic Classes

Template dan generic Classes hanya digunakan di satu fungsi saja.

```
template<class T>
class InvalidInputException : public BaseException {
private:
    T Input;
public:
    InvalidInputException(T Input) {
        this->Input = Input;
    }
    virtual void printMessage(){
        cout << "Input " << Input << " tidak valid" << endl;
    }
};
```

Exception berikut digunakan untuk input yang salah, karena input bisa dalam bentuk string atau integer, maka template dan generic Classes dapat diaplikasikan di fungsi ini.

Template Classes juga digunakan pada implementasi STL, contohnya implementasi STL Pair, Vector, dan Array.

2.4 Exception

Exceptions adalah sebuah konsep sederhana yang sangat bermanfaat, dengan menggunakan exception, kita dapat memberhentikan kode kita jika terjadi sebuah skenario yang tidak diinginkan. Kelebihan exception jika dibandingkan dengan sebuah “resolve in function” adalah implementasi dari exception tersebut dapat diletakkan diluar kode sehingga Code Readability meningkat. Juga terdapat keuntungan dimana beberapa fungsi berbeda dapat menggunakan exception yang sama, sehingga tidak terdapat duplikasi kode.

Dalam program ini, Exceptions dibuat secara spesifik untuk menangani satu tipe exception dan lalu dikumpulkan berdasarkan atribut, misalnya sebuah exception yang tidak memiliki atribut dikumpulkan dengan exception sejenisnya, exception yang memiliki atribut dikumpulkan dengan yang sejenisnya. Keuntungan dari desain kode seperti ini adalah karena exceptions dibuat secara spesifik, penggunaan dan tujuannya di program lainnya terutama main.cpp sangat jelas. Tetapi ada kelemahannya adalah karena exceptions dibuat terpisah dan dikumpulkan berdasarkan atribut, sebuah exception inventory dapat digabung dengan exception crafting sehingga dapat sulit untuk mencari exception inventory saja.

Kelebihan : Jelas dan mudah dimengerti di program lain

Kekurangan : Exception.hpp lebih sulit untuk dimengerti dan di maintain.

2.4.1 Base Exception

Ini adalah Exception dasar yang tidak memiliki atribut lain dan hanya memiliki 1 error message yang akan selalu sama.

```cpp
class BaseException {
public:
    // menuliskan pesan kesalahan ke stdout
    virtual void printMessage() = 0;
};

class InventoryFull : public BaseException {
public:
    public:
    void printMessage() {
        cout << "Inventory penuh!" << endl;
    }
};
class NoRecipe : public BaseException {
public:
    void printMessage() {
        cout << "Tidak ada Recipe yang valid" << endl;
    }
};
class ItemNotFound : public BaseException {
public:
    void printMessage() {
        cout << "Item tidak ditemukan" << endl;
    }
};
class ToolStackingException : public BaseException {
    public:
    virtual void printMessage(){
        cout << "Tidak Bisa Melakukan Stacking pada Tool." <<endl;
    };
};
```

2.4.2 Attributed Exceptions

Ini adalah exceptions yang memiliki sebuah atribut dasar yang akan digunakan dalam error message yang dikeluarkan oleh exception.Tiap Exception dapat memiliki atribut berbeda seperti InvalidNumberException akan menggunakan atribut integer. Terdapat juga InvalidInputException yang menggunakan generic class sehingga bisa menerima berbagai error input baik error integer, string, maupun lainnya.

```
template<class T>
class InvalidInputException : public BaseException {
private:
    T Input;
public:
    InvalidInputException(T Input) {
        this->Input = Input;
    }
    virtual void printMessage(){
        cout << "Input " << Input << " tidak valid" << endl;
    }
};
class InvalidNumberException : public BaseException {
private:
    int number;
public:
    InvalidNumberException(int number) {
        this->number = number;
    }
    void printMessage() {
        cout << "Bilangan " << number << " tidak valid" << endl;
    }
};
class InputGreaterException : public BaseException {···
class ItemNameException : public BaseException {···
class DifferentItemStackException : public BaseException {···
```

2.4.3 Slot ID Exceptions

Karena terdapat berbagai exception untuk menghandle error yang berbasis ID baik di inventory dan crafting, dibuat sebuah Kelas SlotIdException yang merupakan turunan dari BaseException yang mengandung atribut ID dalam integer. Lalu terdapat banyak exception yang merupakan turunan dari SlotIdException.

```
class SlotIdException : public BaseException {
protected:
    int SLOT_ID;
public:
    SlotIdException(int SLOT_ID) {
        this->SLOT_ID = SLOT_ID;
    }
    virtual void printMessage() = 0;
};
class SlotIdFullException : public SlotIdException{
public:
    SlotIdFullException(int SLOT_ID) : SlotIdException(SLOT_ID) {}
    void printMessage() {
        cout << "INVENTORY SLOT ID: " << SLOT_ID << " sudah full" << endl;
    }
};
class NotToolException : public SlotIdException {
public:
    NotToolException(int SLOT_ID) : SlotIdException(SLOT_ID) {}
    void printMessage() {
        cout << "INVENTORY SLOT ID: " << SLOT_ID << " bukan Tool" << endl;
    }
};
class NotNonToolException : public SlotIdException {
public:
    NotNonToolException(int SLOT_ID) : SlotIdException(SLOT_ID) {}
    void printMessage() {
        cout << "INVENTORY SLOT ID: " << SLOT_ID << " bukan Non-Tool" << endl;
    }
};
class NoItemInventoryException : public SlotIdException {···
class NoItemCraftingException : public SlotIdException {···
```

2.5 C++ Standard Template Library

2.5.1 Pair

```
pair<Item*, int> inventory[27];
```

STL Pair diimplementasikan dalam class Inventory untuk menyimpan Item dan quantity dari item tersebut. STL Pair ini digunakan karena sesuai dengan kebutuhan yaitu dibutuhkan sebuah struktur data yang dapat menyimpan dua data untuk setiap elemen array dan dapat diakses dengan mudah

```
std::pair<Item *, int> table[3][3];
```

STL Pair juga digunakan di class CraftState untuk menyimpan Item dan quantity yang akan digunakan di Crafting dan MultiCrafting. Kuantitas dari item perlu disimpan karena fitur MultiCrafting yang mengharuskan disimpannya kuantitas tiap item.

2.5.2 Vector

```
vector<Recipe> recipes;
vector<Item*> items;
```

STL Vector digunakan dalam class AllConfig untuk menyimpan resep dan item-item di config. STL Vector ini digunakan untuk memudahkan proses penambahan menggunakan method push_back dan pencarian resep-resep dan item-item menggunakan STL Algorithm find.

2.5.3 Array

```
array<int, 2> top_lft;
array<int, 2> bot_rht;
```

STL Array digunakan di dalam class CraftState untuk menyimpan koordinat paling kiri atas dan kanan bawah dari item-item di Tabel CraftState. Alasan digunakannya STL Array ini digunakan sebagai pengganti array biasa untuk menghindari alokasi dan dealokasi yang dapat menyebabkan kebocoran memori ketika tidak di-manage dengan benar.

2.5.4 Algorithm Find

```
int AllConfig::search_recipe_idx(CraftState &cs) const {
  return find(this->recipes.begin(), this->recipes.end(), cs) - this->recipes.begin();
}

int AllConfig::search_item_idx(string name) const {
  return find(this->items.begin(), this->items.end(), name) - this->items.begin();
}
```

STL Algorithm Find digunakan di class AllConfig untuk membantu pencarian resep dan item dari Config yang sesuai dengan kondisi Craftstate. STL Algoritma ini digunakan dengan cara membuat operator overload == untuk pasangan class Recipe dan CraftState dan pasangan class Item dan string..

2.6 Konsep OOP lain

Konsep Abstract Base Class digunakan dalam Exception handling dalam BaseException yang adalah pure virtual function. Alasan digunakannya Abstract Base Class pada kelas BaseException adalah karena pada program kami terdapat banyak exception yang dibuat tergantung pada kasus yang ada.Sehingga dengan diterapkan konsep Abstract Base Class kelas exception yang banyak tersebut lebih fleksibel dan modular.

3. Bonus Yang dikerjakan

3.1 Bonus yang diusulkan oleh spek

3.1.1 Multiple Crafting

```
std::pair<Item *, int> table[3][3];
int mul;
```

Multiple Crafting dilakukan dengan cara menyimpan kuantitas dari item di tabel Crafting. Saat melakukan Craft, akan dicari kuantitas terbanyak yang cukup untuk melakukan Crafting yang akan disimpan ke atribut mul, kemudian jumlah Output akan dikali dengan mul. Setiap item di tabel crafting juga akan dikurangi sebanyak mul dan jika kuantitasnya menjadi 0, item tersebut dihapus.

3.1.2 Item dan Tool Baru

Kami menambahkan 4 item baru, Diamond helmet, chestplate, leggings, boots, pada dasarnya item keempat item tersebut hanya sebuah flavor tambahan dan bekerja seperti tool walaupun seharusnya tidak bisa seratus persen mirip dengan tool.

3.2 Bonus Kreasi Mandiri

3.2.1 Align Output

Fungsi show di Crafting dan Inventory disejajarkan menggunakan setw. Fungsi setw digunakan untuk menetapkan *field width* output. Output pertama-tama dihitung panjangnya (Panjang string untuk nama dan banyak digit untuk durability atau quantity). Output kemudian akan di set sehingga panjang kolomnya sama.

4. Pembagian Tugas

Modul (dalam poin spek)	Implementer	Tester
Item	13520006	13520015, 13520093, 13520135, 13520165
Inventory	13520093	13520006, 13520015, 13520135, 13520165
Crafting	13520135	13520006, 13520015, 13520093, 13520165
Show	13520015, 13520135	13520006, 13520093, 13520165
Give	13520015	13520006, 13520093, 13520135, 13520165
Discard	13520093	13520006, 13520015, 13520135, 13520165
Move	13520006, 13520093, 13520165	13520015, 13520135
Use	13520006	13520015, 13520093, 13520135, 13520165
Craft	13520135	13520006, 13520015, 13520093, 13520165
Export	13520165	13520006, 13520015, 13520093, 13520135
Load	13520165	13520006, 13520015, 13520093, 1352013
Exception	13520015	13520006, 13520093, 13520135, 13520165

5. Lampiran

5.1. Form Asistensi

Kelas : 03
Nomor Kelompok : 03
Nama Kelompok : OOPs

1. 13520006 / Vionie Novencia Thanggestyo
2. 13520015 / Jaya Mangalo Soegeng Rahardjo
3. 13520093 / Vincent Ho
4. 13520135 / Muhammad Alif Putra Yasa
5. 13520165 / Ghazian Tsabit Alkamil

Asisten Pembimbing : Fariz Rizki Ekananda

1. Konten Diskusi

kelompok kita bingung di penggunaan generic function atau generic class? sebenarnya kalau kita pake STL sudah termasuk generic class, atau bisa dibuat array of tool/nontool.

penggunaan command USE ? sebenarnya iya cuma buat ngurangin durability, tapi perlu diingat kalau durability jadi nol maka tool nya akan hilang (berarti perlu dibikin dtor ?)

input file konfigurasi ? nama file nya harusnya gak berubah soalnya item nya udah segitu-gitu aja

command MOVE nomor 5, angka 1 nya itu apa , soalnya harusnya bisa banyak? kayanya buat tanda aja, nanti didiskusikan lagi sama asistensi, angka 1 itu jumlah destinasi

kalau bonus di crafting bisa di tumpuk juga

tips tips buat ngerjain GUI? sebenernya agak straight forward aja

kalau bikin GUI di akhir bakal ngubah banyak kode nya gak ? rekomendasi GUI yang sudah ada library nya di c++ SFML

class Diagram notasinya ikutin yang di matkul RPL

biasanya banyak salah dimana ? masih belum terlalu familiar dengan SOLID dan semacamnya, jadi kadang kadang masih ada repeat thing, misal ada class yang nge-inherit tapi ada method yang gak digunain, dan lain lain.

2. Screenshot Bukti