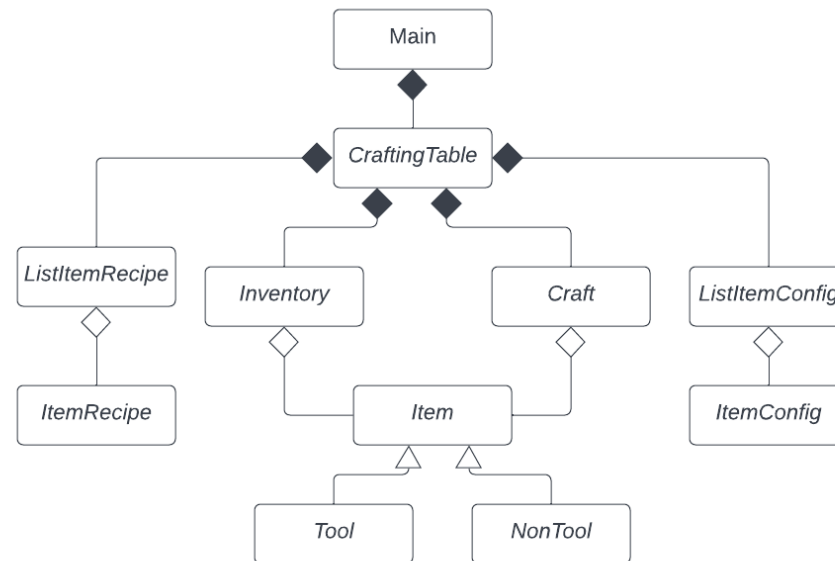


Kelas : 02
Nomor Kelompok : 09
Nama Kelompok : miripmenkrep
1. 13520011 / Muhammad Akyas David Al Aleey
2. 13520014 / Muhammad Helmi Hibatullah
3. 13520026 / Muhammad Fajar Ramadhan
4. 13520116 / Mahesa Lizardy
5. 13520155 / Jundan Haris
Asisten Pembimbing : Michel Fang

1. Diagram Kelas



Gambar 1.1 Diagram Kelas

No	Kelas	Atribut dan Method
1	CraftingTable	<p>Atribut:</p> <ul style="list-style-type: none"> - inv : Inventory - crf : Craft - listItemConfig : ListItemConfig - listRecipeConfig : ListRecipe - command : string - configPath : string - itemConfigPath : string - recipeConfigPath : string <p>Method:</p> <ul style="list-style-type: none"> + CraftingTable() + ~CraftingTable() + readConfig() + readCommand() - readItemConfig() - readItemRecipe() - help() - show() - give() - discard() - move() - moveToCraft() - moveToStack() - moveToInventory() - use()

		<ul style="list-style-type: none"> - craft() - exportInventory()
2	ItemRecipe	<p>Atribut :</p> <ul style="list-style-type: none"> - row : int - col : int - matrix : string[][] - temCraft : string - quantity : int <p>Method:</p> <ul style="list-style-type: none"> + ItemRecipe() + ~ItemRecipe + setRow() + getRow() + setCol() + getCol() + new_matrix() + set_matrix() + getElement() + set_quantity() + get_quantity() + set_item() + get_item() + printRecipe()
3	Inventory	<p>Atribut :</p> <ul style="list-style-type: none"> - slot : vector<Item*> <p>Method:</p> <ul style="list-style-type: none"> + Inventory() + ~Inventory()

		+ operator[]() + isIn() + isFull() + give() + discard() + use()
4	Item	Atribut : # Id : int # name : string # type : string Method: + Item() + ~Item() + operator= + get_id() + get_type() + set_id() + set_name() + set_type() + printInfo() + printExport() + isA()
5	ItemConfig	Atribut : - id : int - name : string - type : string - category : string Method: + ItemConfig()

		<pre> + print() + set_id() + get_id() + set_name() + get_name() + set_type() + get_type() + set_category(); + get_category(); </pre>
6	Craft	<p>Atribut :</p> <ul style="list-style-type: none"> - slot : vector<Item*> - curCraft : string[][] - optCraft : string[][] - namedBased : bool - toolInSlot : int - optRow : int - optCol : int <p>Method:</p> <pre> + Craft() + ~Craft() + isFull() + emptyingCraft() + isRecipe() + emptyRow() + emptyCol() + swapCol() + operator[]() + updateCurCraft() + updateOptimizedCrft() </pre>

7	ListRecipe	Atribut : - listRecipe : ItemRecipe[] - Neff : int Method: + ListRecipe() + ~ListRecipe() + addRecipe() + printListRecipe() + get_Neff() + operator[]()
8	Tool	Atribut : - durability : int Method: + Tool() + ~Tool() + operator=() + get_durability() + set_durability() + use() + operator+() + printInfo() + printExport()
9	NonTool	Atribut : - quantity : int Method: + NonTool() + ~NonTool() + operator=()

		+ get_quantity() + set_quantity() + operator+() + printInfo() + printExport()
10	ListItemConfig	Atribut : - listconfig : ItemConfig[] - Neff : int Method: + ListItemConfig() + ~ListItemConfig() + addElmt() + printList() + get_Neff() + operator[]()

Desain dipilih dalam pengimplementasian adalah *Class Diagram*. Desain dengan *Class Diagram* dipilih karena cocok untuk merepresentasikan paradigma pemrograman berorientasi objek. *Class Diagram* dapat merepresentasikan kelas pada suatu sistem, baik atribut, operasi, ataupun hubungan antar kelas. Kelebihan penggunaan desain terdapat pada kemampuan dalam melakukan kemampuan pada paradigma pemrograman objek. Kekurangan diagram ini adalah terbatasnya keterhubungan antar objek atau atribut dalam modul yang dibuat. Kendala yang dialami selama mendesain kelas OOP ini adalah penyesuaian desain modul dan kelas terhadap ketentuan dan kebutuhan Tugas Besar. Terdapat banyak penyesuaian yang harus dilakukan selama proses implementasi kelas dan method berlangsung. Hubungan antar modul yang dibuat harus sedemikian rupa sehingga mencegah kode yang mubazir, dekomposisi method yang kurang baik, dan struktur kelas yang rumit dan sulit dipahami.

2. Penerapan Konsep OOP

2.1. Inheritance & Polymorphism

Penggunaan Inheritance pada program ini terdapat pada class Tool dan NonTool yang mewarisi class Item. Tool dan NonTool mempunyai banyak atribut yang sama, sehingga kedua class tersebut dapat digeneralisasi agar tidak *DRY (Don't Repeat Yourself)*. Oleh karena itu, dibuatlah class item sebagai abstraksi dari tool dan nontool. Pada class tool akan ditambahkan atribut durability untuk mencatat durability pada item tool. Sedangkan pada class non tool akan ditambahkan atribut quantity untuk mencatat jumlah dari item non tool.

```
35  class Tool : public Item {  
36      private:  
37          int durability;
```

Gambar 2.1.1 class Tool

```
53  class NonTool : public Item {  
54      private:  
55          int quantity;
```

Gambar 2.1.2 class NonTool

2.2. Method/Operator Overloading

Method operator overloading digunakan pada program ini adalah operator[]. Penggunaan operator ini terdapat pada class ListItemConfig, LlistRecipe, Inventory, dan Craft hal ini bertujuan untuk pengaksesan elemen pada list lebih natural


```
232 ✓ ItemConfig ListItemConfig::operator[](int index){  
233     // sesuai id  
234     return this->listconfig[index-1];  
235     // need exception if out of range  
236 }
```

Gambar 2.2.1 operator[] pada ListItemConfig

```
287 ✓ ItemRecipe ListRecipe::operator[](int index){  
288 ✓     if (index >= this->Neff) {  
289         throw "index out of range list";  
290     }  
291     return this->listRecipe[index];  
292 }
```

Gambar 2.2.2 operator[] pada ListRecipe

```
44     Item* &Craft::operator[](int idx) {  
45         // harus ditambahin exception kalo lebih dari 27  
46         return slot[idx];  
47     }
```

Gambar 2.2.3 operator[] pada Craft

```

14  ✓ Item* &Inventory::operator[](int idx) {      Muhammad
15      // harus ditambahin exception kalo lebih dari 27
16      return slot[idx];
17  }

```

Gambar 2.2.4 operator[] pada Inventory

2.3. Template & Generic Classes

Penggunaan Template terdapat pada method isA() yang terdapat pada class item hal ini bertujuan untuk mengecek apakah item tersebut tergolong dalam class Tool atau class non Tool.

```

template<typename T>
bool isA(){
    return (dynamic_cast<T*>(this) != NULL);
}

```

Gambar 2.3.1 Penggunaan Template pada Inventory

2.4. Exception

Penggunaan *exception* pada program terdapat pada method-method yang mengakses indeks dari list seperti listItemConfig, listItemRecipe, Craft, dll. Hal ini bertujuan agar indeks yang diakses tidak melebihi indeks nilai efektif dari sebuah list. Selain itu exception juga terdapat pada input command jika tidak sesuai dengan tipe variabel yang akan di assign.

```
void ItemRecipe::set_matrix(string line, int row, int col){
    if (row >=this->row || col >this->col){
        throw "index out of range";
    }
    this->matrix[row][col] = line;
}
```

Gambar 2.4.1. Exception pada Method set_matrix

```
void ListItemConfig::addElmt(ItemConfig elemen) {
    if (this->Neff == MAX_LIST_CONFIG) {
        throw "list Item is full";
    } else {
        this->listconfig[this->Neff] = elemen;
    }
    this->Neff++;
}
```

Gambar 2.4.1. Exception pada Method addElmt

```
if(cin.fail()) {
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    throw "Wrong quantity input.";
```

Gambar 2.4.1. Exception pada Input yang Salah

2.5. C++ Standard Template Library

Penggunaan STL vector string pada prosedur CraftingTable di file CraftingTable bertujuan untuk menampung string hasil pembacaan file config yang sudah di split berdasarkan spasi. Hal tersebut bertujuan untuk memudahkan assign value kedalam class item config maupun class resep item. STL vector juga digunakan untuk implementasi slot pada Craft dan Inventory.

```
63 void CraftingTable::readConfig() {  
64     // menampung line yang sudah displit  
65     vector<string> words{};  
66     vector<string> wordsrecipe{};
```

Gambar 2.5.1. Penggunaan Vector<string> pada Method readConfig

```
class Inventory {  
    private:  
        vector<Item*> slot;
```

Gambar 2.5.2. Penggunaan Vector sebagai Atribut pada Inventory

```
class Craft {  
    private:  
        vector<Item*> slot;  
        string **curCraft;  
        string **optCraft;
```

Gambar 2.5.3. Penggunaan Vector sebagai Atribut pada Craft

2.6. Konsep OOP lain

1. Abstract Base Class

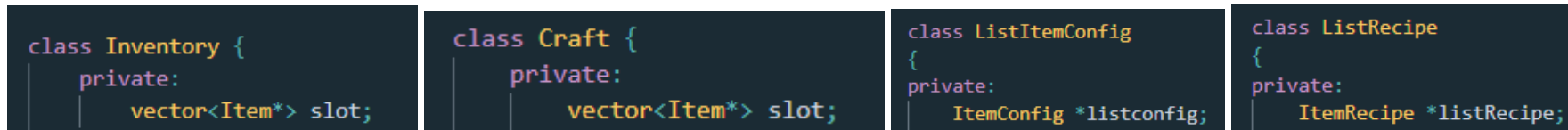
Class Item merupakan penerapan dari Abstract Base Class karena memiliki fungsi *pure virtual* yaitu `printInfo()` dan `printExport()`. Kedua fungsi/prosedur tersebut belum diketahui implementasinya pada *base class* dan akan diimplementasikan di kelas turunannya, yaitu kelas Tool dan NonTool.

```
class Item {
protected:
    int id;
    string name;
    string type;
public:
    Item();
    Item(int id, string name, string type);
    Item(const Item& other);
    Item& operator=(const Item& other);
    ~Item();
    int get_id() const;
    string get_name() const;
    string get_type() const;
    void set_id(int id);
    void set_name(string name);
    void set_type(string type);
    virtual void printInfo()=0;
    virtual string printExport()=0;
    template<typename T>
    bool isA(){
        return (dynamic_cast<T*>(this) != NULL);
    }
};
```

Gambar 2.5.4. Implementasi Abstract Base Class pada Class Item

2. Agregation

Class Inventory, Craft, ListItemConfig, dan ListRecipe merupakan penerapan dari konsep Agregasi. Class Inventory dan Craft menampung kumpulan item, ListItemConfig menampung kumpulan item pada config, dan ListRecipe menampung kumpulan resep. Walaupun kelas Inventory, Craft, ListItemConfig, atau ListRecipe dihapus, elemen-elemen yang ditampung ini masih dapat berdiri sendiri.



```
class Inventory {
private:
    vector<Item*> slot;
}

class Craft {
private:
    vector<Item*> slot;
}

class ListItemConfig
{
private:
    ItemConfig *listconfig;
}

class ListRecipe
{
private:
    ItemRecipe *listRecipe;
}
```

Gambar 2.5.5. Implementasi Aggregation pada Class Inventory, Craft, ListItemConfig, dan ListRecipe

3. Composition

Dua buah kelas dapat dikatakan memiliki hubungan composition jika kelas A mengandung kelas B sehingga kelas A tidak dapat hidup tanpa kelas B. Pada kasus ini, kelas CraftingTable mengandung 4 kelas lain yaitu kelas Inventory, Craft, ListItemConfig, dan ListRecipe.



```
class CraftingTable {
private:
    Inventory inv;
    Craft crf;
    ListItemConfig listItemConfig;
    ListRecipe listRecipeConfig;

    string command;
    string configPath;
    string itemConfigPath;
    string recipeConfigPath;
}
```

Gambar 2.5.6. Implementasi Composition pada Class CraftingTable

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Item dan Tool Baru

Berikut adalah item-item baru yang kami tambahkan, Item dengan ID 26-38 yang merupakan Tool juga ditambahkan konfigurasi recipe untuk item tersebut

NO	ID	Name	Type	TOOL / NONTOL
1	25	GOLD_INGOT	-	NONTOL
2	26	GOLDEN_PICKAXE	-	TOOL
3	27	GOLDEN_AXE	-	TOOL
4	28	GOLDEN_SWORD	-	TOOL
5	29	WOODEN_SHOVELS	-	TOOL
6	30	STONE_SHOVELS	-	TOOL
7	31	IRON_SHOVELS	-	TOOL
8	32	GOLDEN_SHOVELS	-	TOOL
9	33	DIAMOND_SHOVELS	-	TOOL
10	34	WOODEN_HOE	-	TOOL
11	35	STONE_HOE	-	TOOL
12	36	IRON_HOE	-	TOOL

13	37	GOLDEN_HOE	-	TOOL
14	38	DIAMOND_HOE	-	TOOL

3.2. Bonus Kreasi Mandiri

1. Menambahkan ASCII Art yang ditampilkan ketika memulai program.

Untuk memperindah tampilan saat memulai program :D.



Gambar 3.2.1 ASCII Art saat Memulai Program

2. Command "HELP"

Untuk menampilkan daftar command yang dapat digunakan pengguna.


```

Enter 'HELP' to show the command list.
>>> HELP
COMMAND LIST
=====
HELP      : to show the command list.
ITEMS     : to show the item listed from the config file.
RECIPE    : to show the recipe to craft an item.
SHOW      : to show the inventory and crafting table.
GIVE      : to add an item to inventory.
            for NonTools: GIVE <ITEM_NAME> <ITEM_QTY>
            for Tools   : GIVE <ITEM_NAME>
DISCARD   : to decrease a NonTool item from inventory.
            syntax: DISCARD I<INVENTORY_SLOT_ID> <ITEM_QTY>
MOVE      : to move items between inventory slots or between inventory and crafting table slots.
            inventory > inventory      : MOVE I<INVENTORY_SLOT_ID_SRC> 1 I<INVENTORY_SLOT_ID_DEST>
            inventory > crafting table: MOVE I<INVENTORY_SLOT_ID> N C<CRAFTING_SLOT_ID_1> ... C<CRAFTING_SLOT_ID_N>
            crafting table > inventory: MOVE C<CRAFTING_SLOT_ID> 1 I<INVENTORY_SLOT_ID>
USE       : to use an item (decrease the durability by one).
            syntax: USE I<INVENTORY_SLOT_ID>
CRAFT     : to craft items based on the recipe.
EXPORT    : to export the items inside inventory.
            syntax: EXPORT <FILE_NAME>
EXIT      : to exit the program.

Enter 'HELP' to show the command list.
>>> 

```

Gambar 3.2.2. Tampilan HELP

3. Command "ITEMS"

Untuk menampilkan daftar item pada config.

```

Enter 'HELP' to show the command list.
>>> ITEMS
ID | ITEM NAME | ITEM TYPE | CATEGORY
-----
1 | OAK_LOG | LOG | NONTOL
2 | SPRUCE_LOG | LOG | NONTOL
3 | BIRCH_LOG | LOG | NONTOL
4 | OAK_PLANK | PLANK | NONTOL
5 | SPRUCE_PLANK | PLANK | NONTOL
6 | BIRCH_PLANK | PLANK | NONTOL
7 | STICK | - | NONTOL
8 | COBBLESTONE | STONE | NONTOL
9 | BLACKSTONE | STONE | NONTOL
10 | IRON_INGOT | - | NONTOL
11 | IRON_NUGGET | - | NONTOL
12 | DIAMOND | - | NONTOL
13 | WOODEN_PICKAXE | - | TOOL
14 | STONE_PICKAXE | - | TOOL
15 | IRON_PICKAXE | - | TOOL
16 | DIAMOND_PICKAXE | - | TOOL
17 | WOODEN_AXE | - | TOOL
18 | STONE_AXE | - | TOOL
19 | IRON_AXE | - | TOOL
20 | DIAMOND_AXE | - | TOOL
21 | WOODEN_SWORD | - | TOOL
22 | STONE_SWORD | - | TOOL
23 | IRON_SWORD | - | TOOL
24 | DIAMOND_SWORD | - | TOOL
25 | GOLD_INGOT | - | NONTOL
26 | GOLDEN_PICKAXE | - | TOOL
27 | GOLDEN_AXE | - | TOOL
28 | GOLDEN_SWORD | - | TOOL
29 | WOODEN_SHOVELS | - | TOOL
30 | STONE_SHOVELS | - | TOOL
31 | IRON_SHOVELS | - | TOOL
32 | GOLDEN_SHOVELS | - | TOOL
33 | DIAMOND_SHOVELS | - | TOOL
34 | WOODEN_HOE | - | TOOL
35 | STONE_HOE | - | TOOL
36 | IRON_HOE | - | TOOL
37 | GOLDEN_HOE | - | TOOL
38 | DIAMOND_HOE | - | TOOL

```

Gambar 3.2.3. Tampilan ITEMS

4. Command "RECIPE"

Untuk menampilkan daftar resep yang dapat digunakan pada crafting.

```

Enter 'HELP' to show the command list.
>>> RECIPE
1. Name: BIRCH_PLANK (4)
   Row: 1 Col: 1
   BIRCH_LOG
-----
2. Name: DIAMOND_AXE (1)
   Row: 3 Col: 2
   DIAMOND DIAMOND
   DIAMOND STICK
   - STICK
-----
3. Name: DIAMOND_HOE (1)
   Row: 3 Col: 2
   DIAMOND DIAMOND
   - STICK
   - STICK
-----
4. Name: DIAMOND_PICKAXE (1)
   Row: 3 Col: 3
   DIAMOND DIAMOND DIAMOND
   - STICK -
   - STICK -
-----
5. Name: DIAMOND_SHOVELS (1)
   Row: 3 Col: 1
   DIAMOND
   STICK
   STICK
-----
6. Name: DIAMOND_SWORD (1)

```

Gambar 3.2.4. Tampilan RECIPE

5. Class CraftingTable

Class untuk menampung command sekaligus implementasi dari command-command yang ada.

4. Pembagian Tugas

Modul (dalam poin spek)	Implementer	Tester
Item	13520014, 13520026	13520014, 13520026
Inventory	13520011, 13520014	13520011, 13520014
Crafting	13520026	13520026
Operasi dan Command		
SHOW	13520014	13520014
GIVE	13520011	13520011
DISCARD	13520011	13520011
MOVE	13520155	13520155, 13520014
USE	13520014	13520014
CRAFT	13520026	13520026
EXPORT	13520116	13520116
Alur Program dan Konfigurasi		
CraftingTable	13520011, 13520014, 13520026, 13520116, 13520155	13520011, 13520014, 13520026, 13520116, 13520155
Konfigurasi Item	13520116	13520116
Konfigurasi Resep	13520116	13520116

