

Kelas : 02
 Nomor Kelompok : 01
 Nama Kelompok : BRISUPREMACY
 1. 13520059 / Suryanto
 2. 13520071 / Wesly Giovano
 3. 13520092 / Vieri Mansyl
 4. 13520113 / Brianaldo Phandiarta
 5. 13520131 / Steven
 6. 13520167 / Aldwin Hardi Swastia
 Asisten Pembimbing : Gregorius Jovan Kresnadi

1. Diagram Kelas

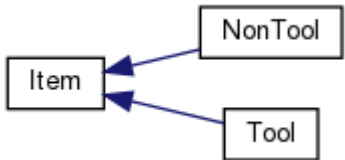
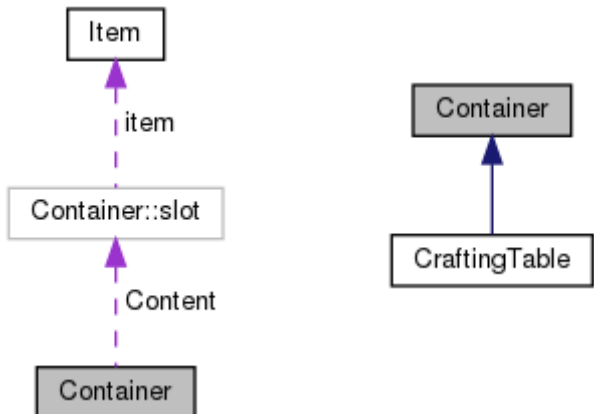
Diagram kelas dari program hasil Tugas Besar 1 yang kami rancang adalah sebagai berikut.

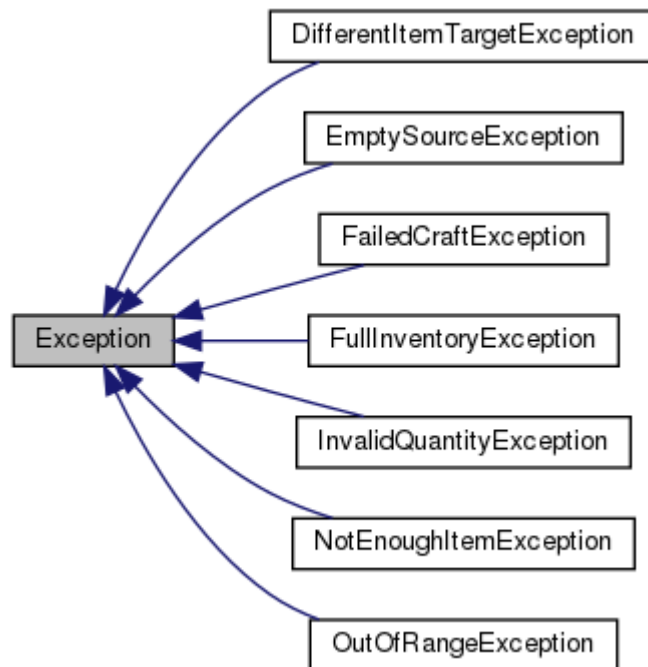
Catatan : +Public
 -Private
 #Protected
static

Nama Kelas	Atribut	Method
Container	#size : int #Content : Slot	+Container() +getItem() : Slot +int getSize() : Int +insert() : void +discard() : void +display() : void <u>+move() : void</u> <u>+swap() : void</u>
CraftingTable		+getName() : string +getType() : string

		+isEmpty() : bool +isTool() : bool +isNonTool() : bool +check() : bool +checkMirror() : bool +checkSub() : bool +craft() : void +what() : void
Exception DifferentItemTargetException EmptySourceException FailedCraftException FullInventoryException InvalidQuantityException NotEnoughItemException OutOfRangeException		
Item	#type : ItemType #id : int #name : string	+item() +getID() : int +getName() : string +getType() : ItemType +getTypeToString() : string +output() : string
NonTool		+NonTool() +operator= () : NonTool +output() : string
Tool	-durability : int	+Tool() +operator= () : Tool +getDurability() : int +repair() : void +use() : void +output() : string
Recipe		+Recipe() +setRow() : void +getRow() : int +setColumn : void +getColumn : int

		+setBlueprint : void +getBlueprint : *String +operator[] : string +setItemName : void +getItemName : string +setCreatedProduct : void +getCreatedProduct : int
--	--	--

Desain Diagram Kelas	Penjelasan
 <pre> classDiagram class Item class NonTool class Tool Item < -- NonTool Item < -- Tool </pre>	<p>Dibentuk kelas Item dikarenakan pada deskripsi tugas, terdapat 2 jenis objek yang dapat digunakan sebagai 'material' dalam proses crafting, yaitu objek Tool dan NonTool yang mana memiliki karakteristik yang sama secara umum.</p> <p>Kelebihan : - Kekurangan : - Kendala yang dialami dari penggunaan kelas Item : -</p>
 <pre> classDiagram class Item class Container class CraftingTable class ContainerSlot["Container::slot"] Container < -- CraftingTable ContainerSlot ..> Item : item Container ..> ContainerSlot : Content </pre>	<p>Dibentuk kelas Container dengan slot berkelas Item dikarenakan pada deskripsi tugas, terdapat objek 'inventory' serta 'crafting table' yang memiliki karakteristik yang mirip, yaitu menampung objek Item sebagai material untuk men-<i>craft</i>. Dengan demikian, objek inventory direalisasikan melalui kelas Container.</p> <p>Objek crafting table merupakan inheritance dari kelas container dikarenakan objek crafting table memiliki beberapa fungsionalitas yang akan diutilisasi selama proses <i>crafting</i> nantinya.</p> <p>Kelebihan : - Kekurangan : - Kendala yang dialami dari penggunaan kelas Item : -</p>



Dibentuk kelas Exception bertujuan untuk mengatasi error yang dapat terjadi ketika user menggunakan program. Untuk mengantisipasi berbagai jenis error, maka dibuat Exception khusus yang merupakan inheritance dari kelas Exception, yaitu sebagai berikut.

- **DifferentItemTargetException**
Meng-*handle* error ketika memindahkan item ke slot yang berisi item yang berbeda
- **EmptySourceException**
Meng-*handle* error ketika user melakukan aktivitas terhadap slot yang kosong (tidak ada item pada slot tersebut)
- **FailedCraftException**
Meng-*handle* error ketika item-item yang berada pada crafting table tidak sesuai dengan seluruh recipe.
- **FullInventoryException**
Meng-*handle* error ketika inventory telah penuh
- **InvalidQuantityException**
Meng-*handle* error ketika user menggunakan command pemanggilan abnormal terhadap jumlah dari item (jumlah Item yang dipanggil < 1)
- **NotEnoughItemException**
Meng-*handle* error ketika user menggunakan command pemanggilan abnormal terhadap jumlah dari item (jumlah Item yang dipanggil > jumlah item yang tersedia)
- **OutOfRangeException**
Meng-*handle* error ketika user menggunakan command pemanggilan abnormal terhadap jumlah indeks dari slot, baik slot pada inventory maupun slot pada crafting table

Kelebihan : -

Kekurangan : -

Kendala yang dialami dari penggunaan kelas Item : -

2. Penerapan Konsep OOP

2.1. Inheritance & Polymorphism

2.1.1. Item

Class Item didefinisikan sebagai Parent Class dan memiliki dua child class yaitu Class Tool dan NonTool. Hal ini dilakukan karena Tool dan NonTool memiliki karakteristik yang sama yaitu sebagai Item. Keuntungan dari menggunakan inheritancenya adalah bisa melakukan reference tool dan nontool dengan satu type sehingga tidak memerlukan 2 buah array dengan tipe berbeda, melainkan hanya memerlukan satu array saja.

```

1  enum class ItemType {None, Log, Plank, Stone, Tool};
2
3  class Item
4  {
5      protected:
6          ItemType type;
7          int id;
8          string name;
9
10     public:
11         Item();
12         Item(int id, string name, ItemType type);
13         int getID();
14         string getName();
15         ItemType getType();
16         string getTypeToString();
17         virtual void output() = 0;
18     };
19
20     extern vector<Item*> listItem;

```

```

1  Tool::Tool(int id, std::string name, int durability) :
2      Item(id, name, ItemType::Tool),
3      durability(durability)
4  {}

```

```

1  NonTool::NonTool(int id, std::string name, ItemType type) :
2      Item(id, name, type)
3  {}

```

2.1.2. Container

Class Container didefinisikan sebagai Parent Class dan memiliki satu Child Class yaitu Class CraftingTable. Hal ini dilakukan karena karakteristik dari Class CraftingTable memiliki kemiripan dengan Class Container. Class CraftingTable memiliki method tambahan yang berfungsi untuk melakukan *crafting*. Keuntungan menggunakan Inheritance pada Class CraftingTable adalah mempermudah untuk melakukan pemindahan item (MOVE) antar-Container.

```

1  class Container {
2      const int MAX_SLOT_QTY = 64;
3
4      typedef struct slot {
5          Item* item;
6          int qty;
7      } Slot;
8
9      protected:
10         int size;
11         Slot* Content;
12
13     public:
14         Container(int size);
15         ~Container();
16         Slot getItem(int index);
17         int getSize();
18         void insert(int n, Item& itemX);
19         void insert(Item& itemX, int durability);
20         void insert(int n, Item& itemX, int index);
21         void discard(int index, int n);
22         void display();
23         static void move(Container& src, int srcIdx, Container& dst, int dstIdx);
24         static void move(Container& src, int srcIdx, Container& dst, int dstIdx, int n);
25 };

```

```

1  CraftingTable::CraftingTable() : Container(9) {}

```

2.2. Method/Operator Overloading

2.2.1. Container

Method overloading digunakan untuk mendukung kebutuhan insert dan move yang membutuhkan parameter yang berbeda. Sebagai penjelasan, pada insert terdapat insert yang digunakan untuk memasukan sejumlah item n (signature: n, Item&), memasukkan item dengan durability tertentu (signature: Item&, int), dan insert sejumlah n item pada indeks tertentu (signature: int, Item&, int). Sedangkan pada move, overloading digunakan untuk mendukung prosedur pemindahan item secara menyeluruh (signature: Container&, int, Container&, int) dan pemindahan item dengan jumlah tertentu (signature: Container&, int, Container&, int, int).

```

1  class Container {
2      const int MAX_SLOT_QTY = 64;
3
4      typedef struct slot {
5          Item* item;
6          int qty;
7      } Slot;
8
9      protected:
10         int size;
11         Slot* Content;
12
13     public:
14         Container(int size);
15         ~Container();
16         Slot getItem(int index);
17         int getSize();
18         void insert(int n, Item& itemX);
19         void insert(Item& itemX, int durability);
20         void insert(int n, Item& itemX, int index);
21         void discard(int index, int n);
22         void display();
23         static void move(Container& src, int srcIdx, Container& dst, int dstIdx);
24         static void move(Container& src, int srcIdx, Container& dst, int dstIdx, int n);
25 };

```

2.2.2. Item

Operator assignment overloading diimplementasikan pada kelas Tool dan NonTool. Hal ini dikarenakan operator assignment overloading memiliki keuntungan berupa menampilkan kode yang lebih jelas dibandingkan dengan menggunakan function. Contohnya adalah `tool.set(otherTool)` lebih sulit dimengerti dibandingkan `tool = otherTool`.

```

1 class Tool : public Item
2 {
3 private:
4     int durability;
5 public:
6     Tool(int id, std::string name, int durability);
7     Tool& operator=(const Tool& other);
8     int getDurability();
9     void repair(int n);
10    void use();
11    string output(int qty) override;
12 };

```

```

1 class NonTool : public Item
2 {
3 public:
4     NonTool(int id, std::string name, ItemType type);
5     NonTool operator=(const NonTool& other);
6     string output(int qty) override;
7 };

```

2.3. Template & Generic Classes

Pada program ini, digunakan STL vector sekaligus diimplementasikan konsep generic class dengan merancang STL vector bertipe pointer dari kelas Item serta pointer dari kelas Recipe. Penggunaan template disesuaikan dengan kebutuhan dari program dikarenakan STL vector diutilisasi untuk menampung Item yang terdiri dari Tool dan Nontool serta recipe-recipe yang tersedia.

```

1 vector<Recipe*> recipes;
2 vector<Item*> listItem;

```


2.4. Exception

Pada program ini sebuah base class Exception yang menurunkan banyak kelas exception khusus yang lain yang digunakan pada program, misalnya EmptySourceException dan FullInventoryException. Alasan penggunaan Exception adalah untuk menangani berbagai kondisi dan kesalahan input yang dapat mengakibatkan error pada program. Keuntungan dari implementasi konsep ini yaitu program tidak akan berhenti secara total ketika terdapat kesalahan, error dapat di-catch dan dimunculkan pesan errornya agar pengguna mengetahui kesalahan yang baru saja terjadi.

```

1  class Exception {
2      public:
3          Exception() {}
4          virtual ~Exception() {}
5          virtual string what() = 0;
6  };
7
8
9  class EmptySourceException : public Exception {
10     public:
11         EmptySourceException() : Exception() {}
12         string what() {
13             return "Source slot is empty. Fails to perform operation.\n";
14         }
15         ~EmptySourceException(){}
16 };
17
18 class FullInventoryException : public Exception {
19     public:
20         FullInventoryException() : Exception() {}
21         string what() {
22             return "Destination is Full. Fails to perform operation.\n";
23         }
24         ~FullInventoryException(){}
25 };
26
27 class DifferentItemTargetException : public Exception {
28     public:
29         DifferentItemTargetException() : Exception() {}
30         string what() {
31             return "Target slot currently contains different item. Fails to perform operation.\n";
32         }
33         ~DifferentItemTargetException() {}
34 };
35

```

```

36 class NotEnoughItemException : public Exception {
37     public:
38         NotEnoughItemException() : Exception() {}
39         string what() {
40             return "Source slot does not have enough item quantity. Fails to perform operation.\n";
41         }
42         ~NotEnoughItemException() {}
43 };
44
45 class InvalidQuantityException : public Exception {
46     public:
47         InvalidQuantityException() : Exception() {}
48         string what() {
49             return "Item quantity is invalid. Fails to perform operation.\n";
50         }
51         ~InvalidQuantityException() {}
52 };
53
54 class FailedCraftException : public Exception {
55     public:
56         FailedCraftException() : Exception() {}
57         string what() {
58             return "Crafting table does not match any recipe. Fails to perform operation.\n";
59         }
60 };

```

2.5. C++ Standard Template Library

2.5.1. Container

Pada Class Container digunakan Vector yang merupakan STL. Hal ini dikarenakan dibandingkan dengan array yang merupakan list statik, vector yang merupakan list dinamis lebih mudah untuk diimplementasikan dan diolah (menggunakan fungsi bawaan dari vector yakni push dan pop). Tidak hanya itu, STL vector juga dapat memberikan error checking dan safety. Misalnya ketika menggunakan .at(), apabila indexnya salah, STL vector akan melakukan throw error sedangkan array biasa tidak dapat melakukan hal tersebut.



```
1 vector<Recipe*> recipes;  
2 vector<Item*> listItem;
```



```
1 std::vector<Recipe*>::iterator ptr = recipes.begin();  
2     bool flag = false;  
3     string item_name;  
4  
5     while (ptr < recipes.end() && !flag) {  
6         if (this->check((*ptr)->getBlueprint()) || this->checkMirror((*ptr)->getBlueprint()) ||  
7             (this->checkSub((*ptr)->getBlueprint(), (*ptr)->getRow(), (*ptr)->getColumn())) {  
8             flag = true;  
9             item_name = (*ptr)->getItemName();  
10        }  
11        else ptr++;  
12    }
```

2.6. Abstract Base Class

2.6.1. Item

Kelas Item diimplementasikan sebagai class abstrak karena untuk setiap child classnya akan memiliki atribut yang berbeda dan method output yang berbeda. Keuntungan dari implementasi konsep ini adalah mempermudah dalam proses output informasi inventory ke suatu file txt.

```
1  enum class ItemType {None, Log, Plank, Stone, Tool};
2
3  class Item
4  {
5      protected:
6          ItemType type;
7          int id;
8          string name;
9
10     public:
11         Item();
12         Item(int id, string name, ItemType type);
13         int getID();
14         string getName();
15         ItemType getType();
16         string getTypeToString();
17         virtual void output() = 0;
18     };
19
20 extern vector<Item*> listItem;
```

2.6.2. Exception

Kelas exception adalah kelas abstrak karena adanya suatu method yang tidak diimplementasikan, yaitu method what. Alasan konsep ini digunakan karena dibutuhkan berbagai komentar yang berbeda untuk tiap exception. Selain itu, Program juga hanya perlu melakukan catch ke satu class yaitu Exception meskipun objek yang di-throw dapat berupa kelas apa saja (inheritance dari class exception).

```

1  class Exception {
2      public:
3          Exception() {}
4          virtual ~Exception() {}
5          virtual string what() = 0;
6  };
7
8
9  class EmptySourceException : public Exception {
10     public:
11         EmptySourceException() : Exception() {}
12         string what() {
13             return "Source slot is empty. Fails to perform operation.\n";
14         }
15         ~EmptySourceException(){}
16 };
17
18 class FullInventoryException : public Exception {
19     public:
20         FullInventoryException() : Exception() {}
21         string what() {
22             return "Destination is Full. Fails to perform operation.\n";
23         }
24         ~FullInventoryException(){}
25 };
26
27 class DifferentItemTargetException : public Exception {
28     public:
29         DifferentItemTargetException() : Exception() {}
30         string what() {
31             return "Target slot currently contains different item. Fails to perform operation.\n";
32         }
33         ~DifferentItemTargetException() {}
34 };
35

```

```

36 class NotEnoughItemException : public Exception {
37     public:
38         NotEnoughItemException() : Exception() {}
39         string what() {
40             return "Source slot does not have enough item quantity. Fails to perform operation.\n";
41         }
42         ~NotEnoughItemException() {}
43 };
44
45 class InvalidQuantityException : public Exception {
46     public:
47         InvalidQuantityException() : Exception() {}
48         string what() {
49             return "Item quantity is invalid. Fails to perform operation.\n";
50         }
51         ~InvalidQuantityException() {}
52 };
53
54 class FailedCraftException : public Exception {
55     public:
56         FailedCraftException() : Exception() {}
57         string what() {
58             return "Crafting table does not match any recipe. Fails to perform operation.\n";
59         }
60 };

```

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Multiple Crafting

Seperti dengan konsep *crafting*, *Multiple crafting* memungkinkan pengguna untuk melakukan *crafting* secara bersamaan dalam sekali proses. *Multiple crafting* diimplementasikan dengan menaikkan jumlah (*amount*) maksimum slot pada crafting table sedemikian sehingga item-item yang sama dapat di-stack di slot tersebut. Item-item yang bertumpuk akan di-*craft* menjadi item baru sesuai dengan recipe yang tersedia. Berikut adalah screenshot cuplikan implementasi bonus ini.

SEBELUM :

NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]

NULL[0]	DIAMOND[5]	NULL[0]
NULL[0]	DIAMOND[5]	NULL[0]
NULL[0]	STICK[5]	NULL[0]

Command : CRAFT
Item successfully crafted !!

Command : SHOW

DIAMOND_SWORD[10]	DIAMOND_SWORD[10]	DIAMOND_SWORD[10]	DIAMOND_SWORD[10]	DIAMOND_SWORD[10]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]

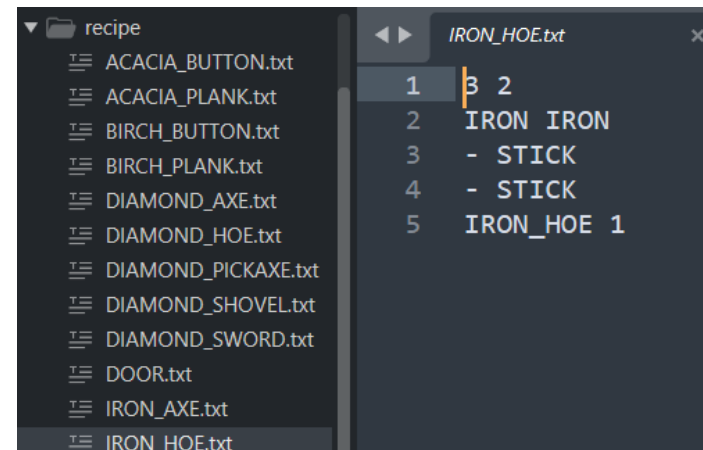
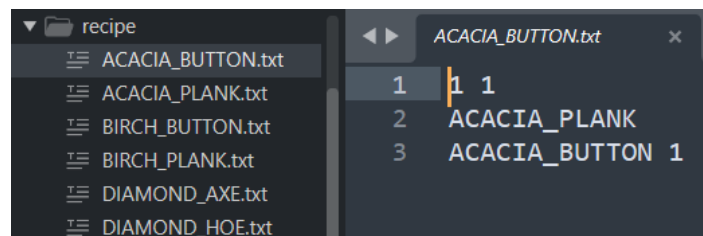
NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]

SETELAH :

3.1.2. Item dan Recipe Baru

Dilakukan penambahan item serta recipe baru yang dapat diutilisasi pada program ini. Penambahan item baru diimplementasikan dengan menambahkan id item, nama item, dan tipe item ke item.txt serta penambahan recipe diimplementasikan dengan menambah recipenya ke folder recipe

Berikut adalah screenshot cuplikan implementasi bonus ini.



```

item.txt
1 1 OAK_LOG LOG NONTOL
2 2 SPRUCE_LOG LOG NONTOL
3 3 BIRCH_LOG LOG NONTOL
4 4 OAK_PLANK PLANK NONTOL
5 5 SPRUCE_PLANK PLANK NONTOL
6 6 BIRCH_PLANK PLANK NONTOL
7 7 STICK - NONTOL
8 8 COBBLESTONE STONE NONTOL
9 9 BLACKSTONE STONE NONTOL
10 10 IRON_INGOT - NONTOL
11 11 IRON_NUGGET - NONTOL
12 12 DIAMOND - NONTOL
13 13 ACACIA_LOG LOG NONTOL
14 14 ACACIA_PLANK PLANK NONTOL
15 15 TORCH - NONTOL
16 16 STRING - NONTOL
17 17 WOOL - NONTOL
18 18 STONE_BUTTON - NONTOL
19 19 OAK_BUTTON - NONTOL
20 20 SPRUCE_BUTTON - NONTOL
21 21 BIRCH_BUTTON - NONTOL
22 22 ACACIA_BUTTON - NONTOL
23 23 DOOR - NONTOL

```

```

item.txt
24 24 BED - NONTOL
25 25 STONE_STAIRS - NONTOL
26 26 WOODEN_STAIRS - NONTOL
27 27 WOODEN_PICKAXE - TOOL
28 28 STONE_PICKAXE - TOOL
29 29 IRON_PICKAXE - TOOL
30 30 DIAMOND_PICKAXE - TOOL
31 31 WOODEN_AXE - TOOL
32 32 STONE_AXE - TOOL
33 33 IRON_AXE - TOOL
34 34 DIAMOND_AXE - TOOL
35 35 WOODEN_SWORD - TOOL
36 36 STONE_SWORD - TOOL
37 37 IRON_SWORD - TOOL
38 38 DIAMOND_SWORD - TOOL
39 39 WOODEN_HOE - TOOL
40 40 STONE_HOE - TOOL
41 41 IRON_HOE - TOOL
42 42 DIAMOND_HOE - TOOL
43 43 WOODEN_SHOVEL - TOOL
44 44 STONE_SHOVEL - TOOL
45 45 IRON_SHOVEL - TOOL
46 46 DIAMOND_SHOVEL - TOOL

```

```

recipe
ACACIA_BUTTON.txt
ACACIA_PLANK.txt
BIRCH_BUTTON.txt
BIRCH_PLANK.txt
DIAMOND_AXE.txt
DIAMOND_HOE.txt
DIAMOND_PICKAXE.txt
DIAMOND_SHOVEL.txt
DIAMOND_SWORD.txt
DOOR.txt
IRON_AXE.txt
IRON_HOE.txt
IRON_INGOT.txt
IRON_NUGGET.txt
IRON_PICKAXE.txt
IRON_SHOVEL.txt
IRON_SWORD.txt
OAK_BUTTON.txt
OAK_PLANK.txt
SPRUCE_BUTTON.txt
SPRUCE_PLANK.txt
STICK.txt
STONE_AXE.txt
STONE_BUTTON.txt
STONE_HOE.txt
STONE_PICKAXE.txt
STONE_SHOVEL.txt
STONE_STAIRS.txt
STONE_SWORD.txt
TORCH.txt
WOODEN_AXE.txt
WOODEN_HOE.txt
WOODEN_PICKAXE.txt
WOODEN_SHOVEL.txt
WOODEN_STAIRS.txt
WOODEN_SWORD.txt

```

3.2. Bonus Kreasi Mandiri

3.2.1. Swap

Swap adalah fitur untuk memindahkan seluruh item di sebuah slot ke slot lain, dan memindahkan seluruh item yang terdapat di slot tersebut kembali ke slot yang pertama. Swap diimplementasikan dengan menukar Item dan Quantity dari atribut Slot antara satu slot dengan slot lain. Berikut adalah screenshot cuplikan implementasi bonus ini.

SEBELUM :

OAK_PLANK[10]	IRON_HOE[10]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]						
NULL[0]	NULL[0]	NULL[0]						
NULL[0]	NULL[0]	NULL[0]						

SETELAH :

Command : SWAP I0 I1
Item successfully swapped !!

Command : SHOW

IRON_HOE[10]	OAK_PLANK[10]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]	NULL[0]

NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]
NULL[0]	NULL[0]	NULL[0]

3.2.2. Help

Help merupakan fitur yang menampilkan command-command beserta format penulisan command tersebut yang dapat digunakan oleh pengguna. Berikut adalah screenshot cuplikan implementasi bonus ini.

```
Command : HELP
-----
COMMAND      ||      FORMAT
-----
GIVE          || GIVE [item name] [item quantity]
SHOW          || SHOW
CRAFT         || CRAFT
MOVE          || MOVE [inventory's index (I0..I26)]      [item quantity]      [crafting table's index (C0..C8)]
MOVE          || MOVE [crafting table's index (C0..C8)]    [item quantity]      [inventory's index (I0..I26)]
MOVE          || MOVE [inventory's index (I0..I26)]      [item quantity]      [inventory's index (I0..I26)]
MOVE          || MOVE [crafting table's index (C0..C8)]    [item quantity]      [crafting table's index (C0..C8)]
SWAP          || SWAP [inventory's index (I0..I26)]      [crafting table's index (C0..C8)]
SWAP          || SWAP [crafting table's index (C0..C8)]    [inventory's index (I0..I26)]
SWAP          || SWAP [inventory's index (I0..I26)]      [inventory's index (I0..I26)]
SWAP          || SWAP [crafting table's index (C0..C8)]    [crafting table's index (C0..C8)]
USE           || USE [inventory's index (I0..I26)]
DISCARD       || DISCARD [inventory's index (I0..I26)]      [item quantity]
DISCARD       || DISCARD [crafting table's index (C0..C8)] [item quantity]
AVAILABLE     || AVAILABLE
HELP          || HELP
QUIT          || QUIT
-----
```

3.2.3. Available Item

Available merupakan fitur untuk menampilkan item yang dapat 'diberikan' langsung dari pengguna ke dalam inventory. Berikut adalah screenshot cuplikan implementasi bonus ini.

```
Command : AVAILABLE
1 OAK_LOG LOG
2 SPRUCE_LOG LOG
3 BIRCH_LOG LOG
4 OAK_PLANK PLANK
5 SPRUCE_PLANK PLANK
6 BIRCH_PLANK PLANK
7 STICK NONE
8 COBBLESTONE STONE
9 BLACKSTONE STONE
10 IRON_INGOT NONE
11 IRON_NUGGET NONE
12 DIAMOND NONE
13 ACACIA_LOG LOG
14 ACACIA_PLANK PLANK
15 TORCH NONE
16 STRING NONE
17 WOOL NONE
18 STONE_BUTTON NONE
19 OAK_BUTTON NONE
20 SPRUCE_BUTTON NONE
21 BIRCH_BUTTON NONE
22 ACACIA_BUTTON NONE
23 DOOR NONE
24 BED NONE
25 STONE_STAIRS NONE
26 WOODEN_STAIRS NONE
27 WOODEN_PICKAXE TOOL
28 STONE_PICKAXE TOOL
29 IRON_PICKAXE TOOL
30 DIAMOND_PICKAXE TOOL
31 WOODEN_AXE TOOL
32 STONE_AXE TOOL
33 IRON_AXE TOOL
34 DIAMOND_AXE TOOL
35 WOODEN_SWORD TOOL
36 STONE_SWORD TOOL
37 IRON_SWORD TOOL
38 DIAMOND_SWORD TOOL
39 WOODEN_HOE TOOL
40 STONE_HOE TOOL
41 IRON_HOE TOOL
42 DIAMOND_HOE TOOL
43 WOODEN_SHOVEL TOOL
44 STONE_SHOVEL TOOL
45 IRON_SHOVEL TOOL
46 DIAMOND_SHOVEL TOOL
```

4. Pembagian Tugas

Modul (dalam poin spek)	Implementer	Tester
Main	13520131	13520059, 13520071, 13520092, 13520113, 13520131, 13520167
Item, Tool, dan NonTool	13520131	13520059, 13520071, 13520092, 13520113, 13520131, 13520167
Container	13520059, 13520071	13520059, 13520071, 13520092, 13520113, 13520131, 13520167
Crafting Table	13520113	13520059, 13520071, 13520092, 13520113, 13520131, 13520167
Input File dan Output File	13520092, 13520167	13520059, 13520071, 13520092, 13520113, 13520131, 13520167
Exception	13520071	13520059, 13520071, 13520092, 13520113, 13520131, 13520167

5. LAMPIRAN – Form Asistensi

Tanggal Asistensi : 14 Maret 2022

Kelas : 02

Nomor Kelompok : 01

Nama Kelompok : BRISUPREMACY

1. 13520059 / Suryanto

2. 13520071 / Wesly Giovano

3. 13520092 / Vieri Mansyl

4. 13520113 / Brianaldo Phandiarta

5. 13520131 / Steven

6. 13520167 / Aldwin Hardi Swastia

Asisten Pembimbing : Gregorius Jovan Kresnadi

5.1 Konten Diskusi

- Bagaimana Implementasi Generic Function dan Generic Class?

Bisa diimplementasikan di command MOVE . coba dikreasi-in.

- Maksud dari dipaksa dari implementasi komponen class bagaimana?

Yang penting ga dibuat-buat. Batasannya gak bisa disebut secara eksplisit yang jelas gak sengaja dibuat hanya untuk implementasinya.

- Bonus 3 yang berupa tool baru dan resepnya, apakah hanya menambahkan .txtnya saja?

Iya, secara singkat kalian tinggal nambahin resepnya.

- Mengenai konfigurasi item yang di-mirror bagusnya dihandle menggunakan program atau bikin konfigurasi resep lagi?

Bagusnya dihandle langsung dari program.

- Perintah Move sebatas inventory ke crafting table (and vice versa) atau bisa inventory ke inventory atau crafting table ke crafting table?

Bisa Inventory ke Inventory dan Table ke table juga, itu bisa jadi ide untuk generic.

- Kalo move barang ke tempat yang occupied, error atau swap?

Pedomannya ikut ke Minecraft langsung, di-swap

5.2 Screenshot Bukti

