

Kelas : 02
Nomor Kelompok : 05
Nama Kelompok : MineHati
1. 13520029 / Muhammad Garebaldhie ER Rahman
2. 13520062 / Rifqi Naufal Abdjul
3. 13520068 / Muhammad Naufal Satriandana
4. 13520101 / Aira Thalca Avila Putra
5. 13520119 / Marchotridyo
Asisten Pembimbing : Jonathan Yudi Gunawan

DAFTAR ISI

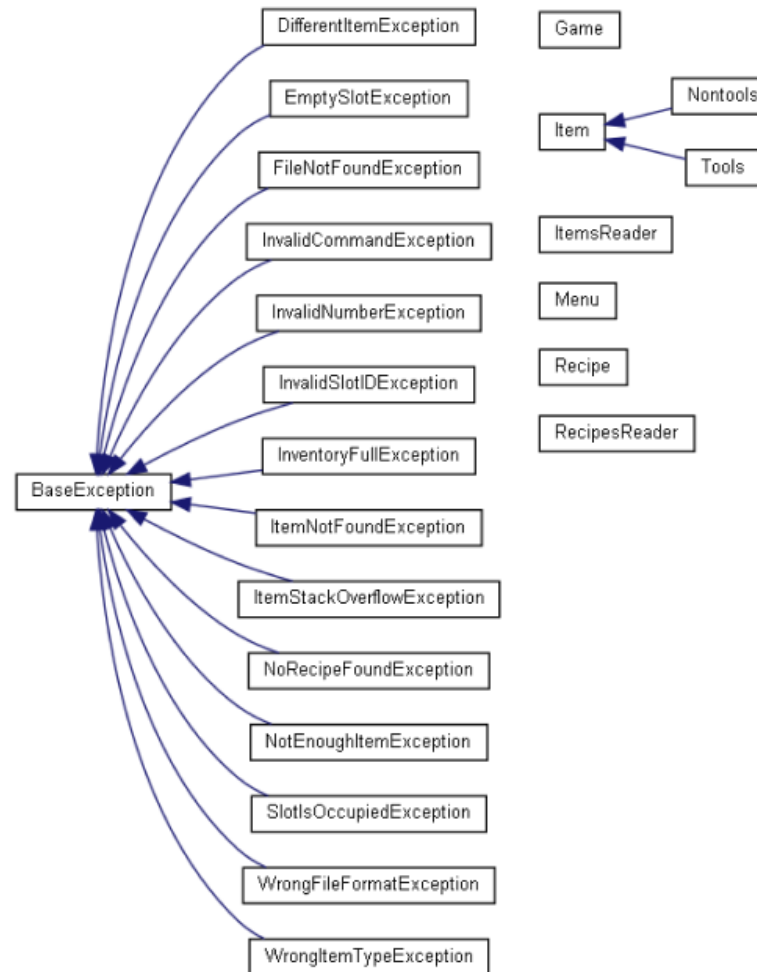
Diagram Kelas	4
1.1 Hierarki Kelas	4
1.2 Kelas BaseException dan Turunannya	5
1.3 Kelas DifferentItemException	6
1.4 Kelas EmptySlotException	7
1.5 Kelas FileNotFoundException	7
1.6 Kelas InvalidCommandException	8
1.7 Kelas InvalidNumberException	8
1.8 Kelas InvalidSlotIDException	9
1.9 Kelas InventoryFullException	9
1.10 Kelas ItemNotFoundException	10

1.11 Kelas ItemStackOverflowException	11
1.12 Kelas NoRecipeFoundException	12
1.13 Kelas NotEnoughItemException	13
1.14 Kelas SlotsOccupiedException	14
1.15 Kelas WrongFormatException	14
1.16 WrongItemTypeException	15
1.17 Kelas Game	16
1.18 Kelas Item, Nontools, dan Tools	17
1.19 Kelas ItemsReader	18
1.20 Kelas Menu	19
1.21 Kelas Recipe	20
1.22 Kelas RecipesReader	20
1.23 Penjelasan Desain Kelas	21
Penerapan Konsep OOP	22
2.1 Inheritance & Polymorphism	22
2.2 Method/Operator Overloading	24
2.3 Template & Generic Classes	26
2.4 Exception	30
2.5 C++ Standard Template Library	33

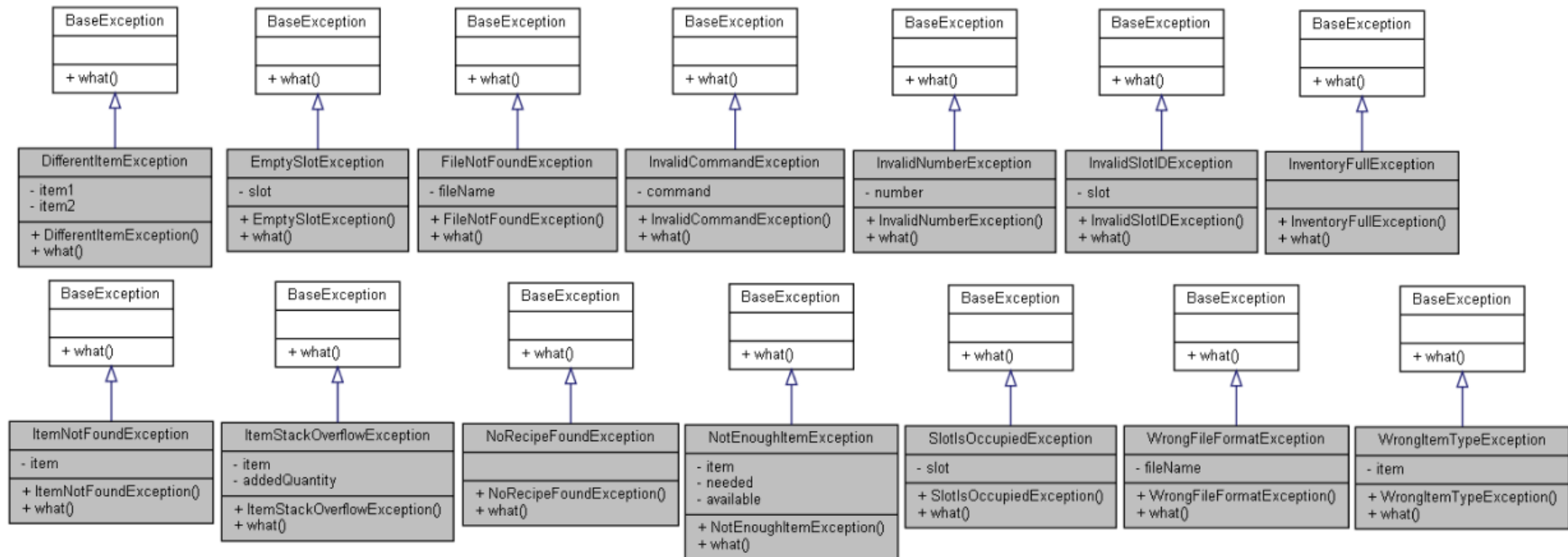
2.6 Abstract Base Class (Konsep OOP Lain)	36
Bonus Yang dikerjakan	38
3.1 Bonus yang diusulkan oleh spek	38
3.1.1 Multiple Crafting	38
3.1.2 Item dan Tool Baru	40
3.1.3 Unit Testing Implementation	41
3.2 Bonus Kreasi Mandiri	48
3.2.1 Command RECIPES	48
3.2.2 Command HELP	53
3.2.3 Setting Directory Konfigurasi File	54
3.2.4 ASCII Art	55
Pembagian Tugas	56

1. Diagram Kelas

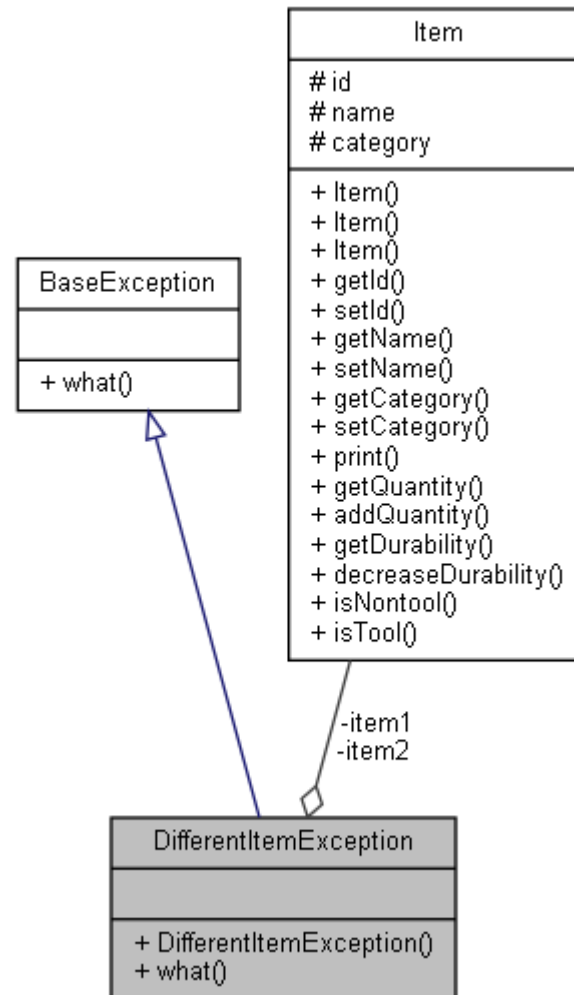
1.1 Hierarki Kelas



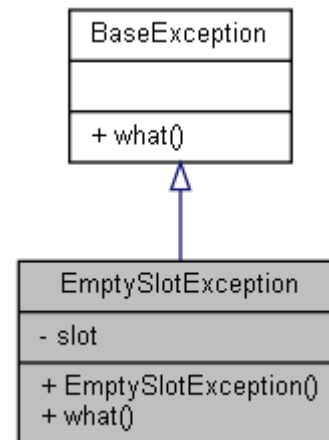
1.2 Kelas BaseException dan Turunannya



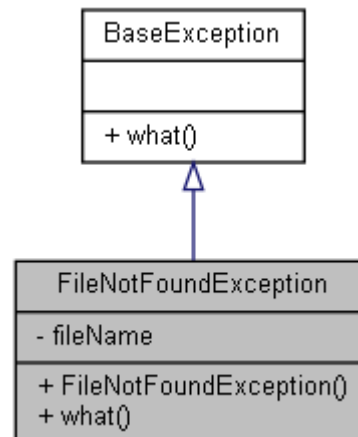
1.3 Kelas DifferentItemException



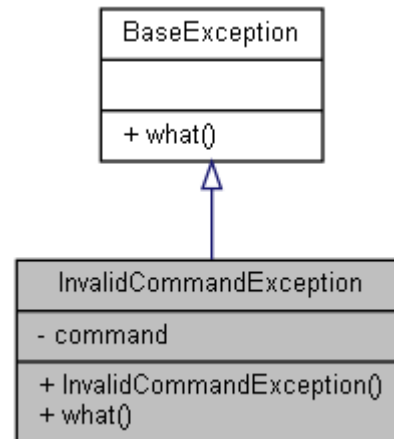
1.4 Kelas EmptySlotException



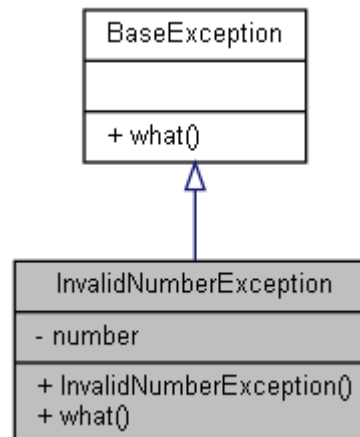
1.5 Kelas FileNotFoundException



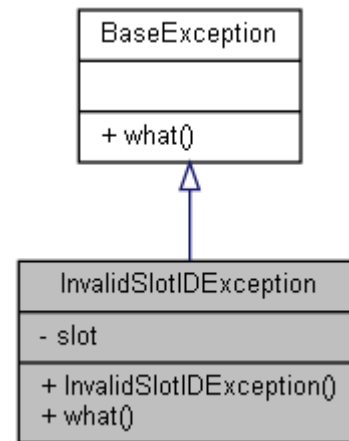
1.6 Kelas InvalidCommandException



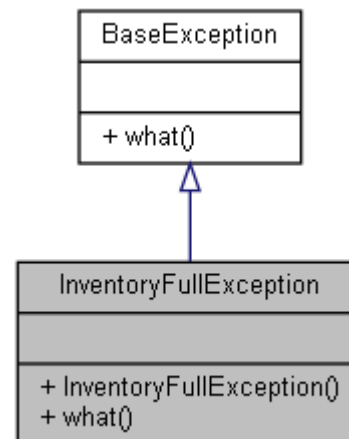
1.7 Kelas InvalidNumberException



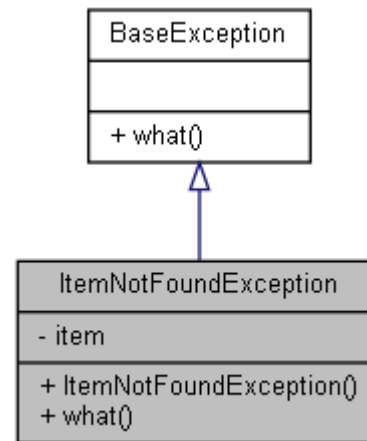
1.8 Kelas InvalidSlotIDException



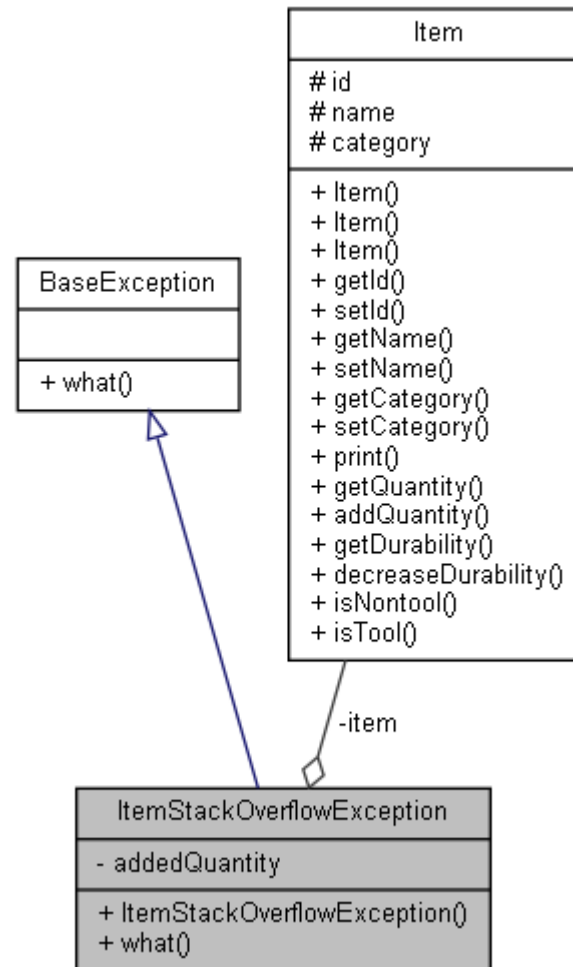
1.9 Kelas InventoryFullException



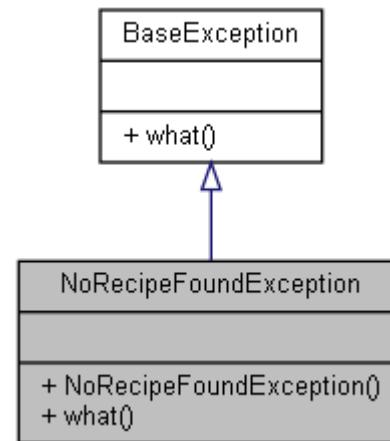
1.10 Kelas ItemNotFoundException



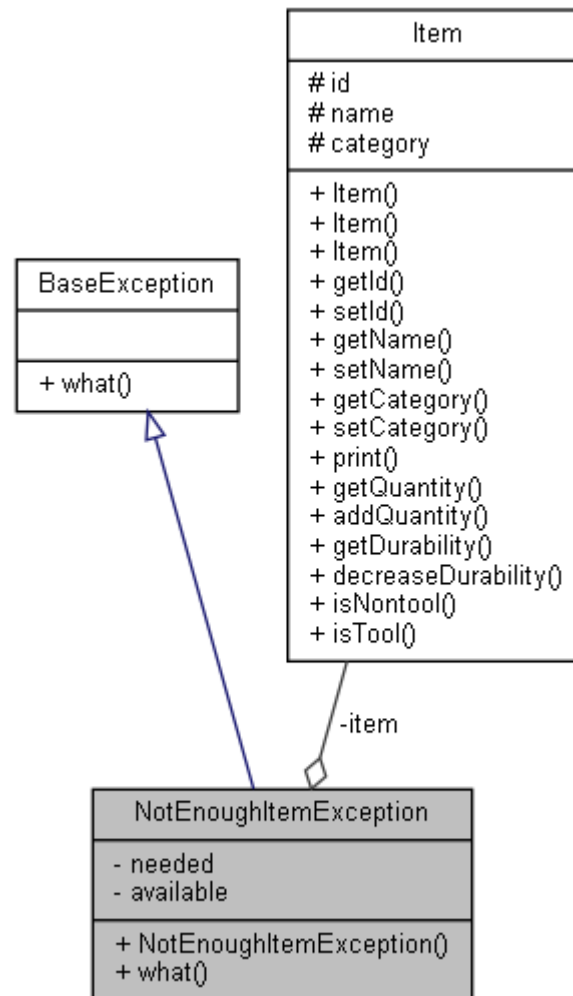
1.11 Kelas ItemStackOverflowException



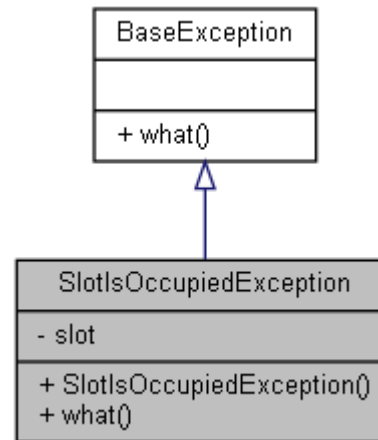
1.12 Kelas NoRecipeFoundException



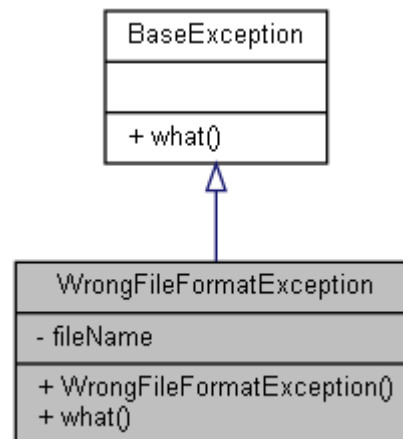
1.13 Kelas NotEnoughItemException



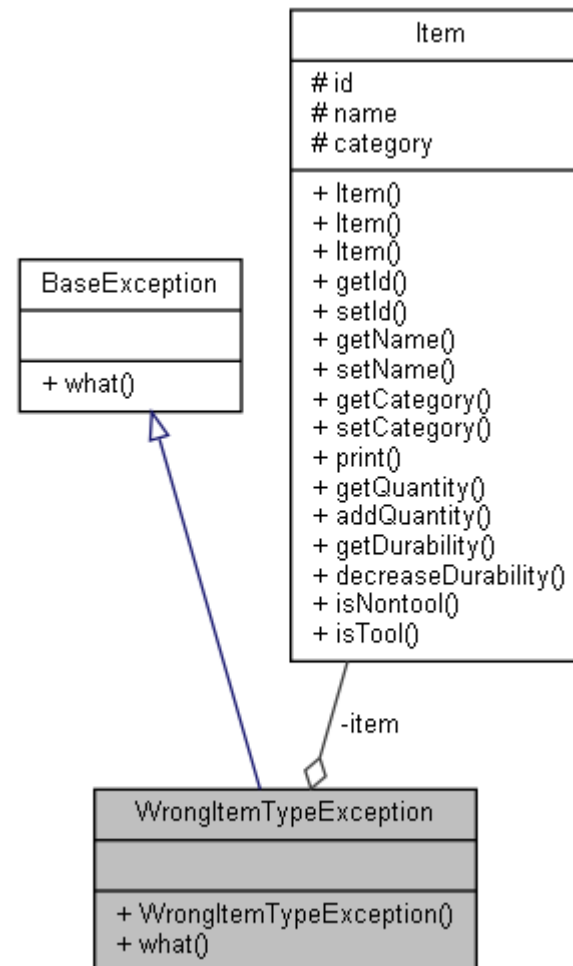
1.14 Kelas SlotsOccupiedException



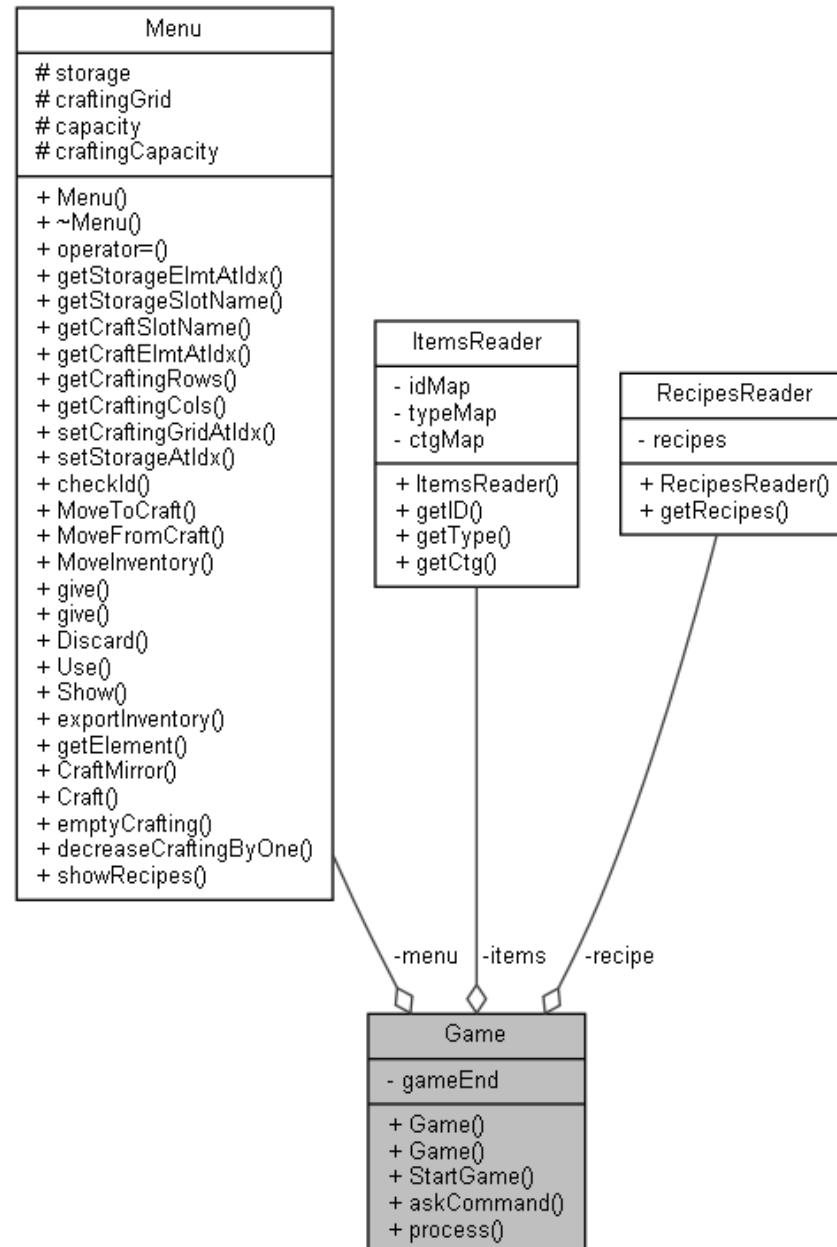
1.15 Kelas WrongFormatException



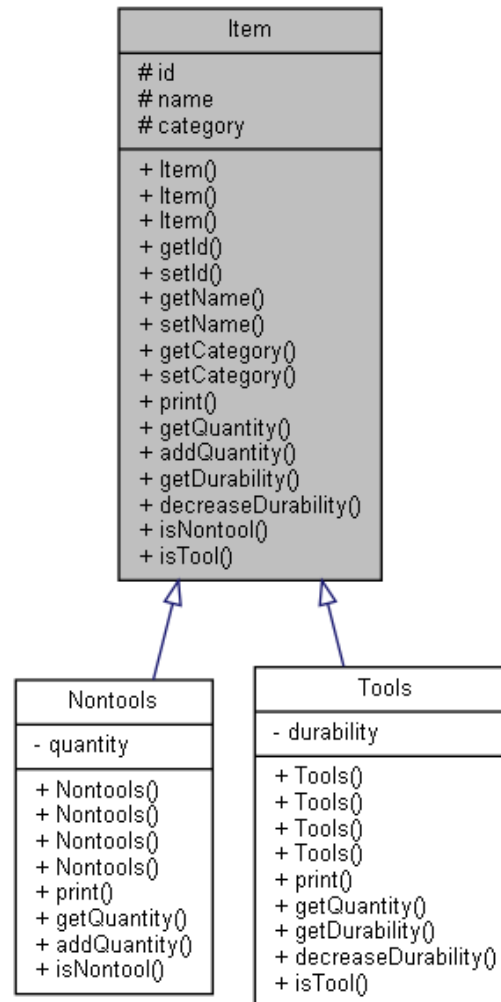
1.16 WrongItemTypeException



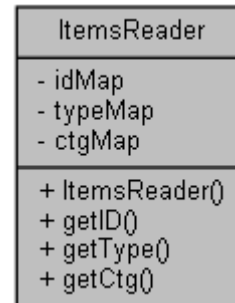
1.17 Kelas Game



1.18 Kelas Item, Nontools, dan Tools



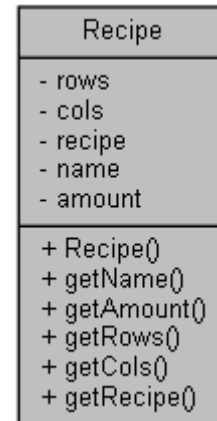
1.19 Kelas ItemsReader



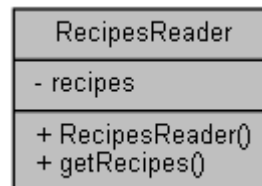
1.20 Kelas Menu

Menu
storage # craftingGrid # capacity # craftingCapacity
+ Menu() + ~Menu() + operator=() + getStorageElmtAtIdx() + getStorageSlotName() + getCraftSlotName() + getCraftElmtAtIdx() + getCraftingRows() + getCraftingCols() + setCraftingGridAtIdx() + setStorageAtIdx() + checkId() + MoveToCraft() + MoveFromCraft() + MoveInventory() + give() + give() + Discard() + Use() + Show() + exportInventory() + getElement() + CraftMirror() + Craft() + emptyCrafting() + decreaseCraftingByOne() + showRecipes()

1.21 Kelas Recipe



1.22 Kelas RecipesReader



1.23 Penjelasan Desain Kelas

Kami memilih untuk mengimplementasikan konsep inheritance pada kelas yang berhubungan dengan Exception dan Item dengan tujuan agar bisa memanfaatkan konsep polymorphism yang disediakan pada C++. Dengan membagi kelas Exception menjadi 1 superclass BaseException dan 14 subclass lainnya (DifferentItemException, EmptySlotException, ...), dalam blok try-catch kita hanya perlu menangkap satu tipe saja, antara pointer ke BaseException ataupun reference ke BaseException. Pendekatan ini akan mengurangi banyak baris kode yang perlu ditulis karena tanpa adanya polymorphism, setiap tipe exception masing-masing harus memiliki handler sendiri.

Begitu pula untuk kelas Item yang dibagi menjadi 1 superclass Item dan 2 subclass, yaitu Nontools dan Tools. Kedua tipe item ini memiliki behaviour yang berbeda namun pada dasarnya keduanya dapat diletakkan pada matriks inventory juga matriks crafting table. Pendekatan ini sangat diperlukan agar kedua matriks tersebut bisa menerima data Item baik dari item yang bertipe Nontools maupun yang bertipe Tools.

Kekurangan dari pendekatan ini adalah karena polymorphism yang digunakan berbasis pointer, kita harus hati-hati dalam mengalokasikan dan mendealokasikan pointer tersebut. Apabila kita tidak hati-hati dan terus mengalokasikan pointer baru tanpa mendealokasikan pointer yang lama, dapat terjadi penumpukan *junk value* pada memori heap.

Kendala yang dialami saat mendesain kelas adalah diperlukan beberapa iterasi untuk membuat struktur final ini. Pada awal pembuatan, belum terlalu tergambarkan masalah apa saja yang akan ditemui dalam proses pemrograman. Seiring pemrograman berjalan, apabila ditemukan suatu masalah yang tidak bisa diselesaikan menggunakan struktur kelas yang sudah didesain sebelumnya, diperlukan perbaikan terhadap struktur kelas lama agar bisa mengakomodasi masalah tersebut. Proses ini dilakukan terus menerus sampai terbentuk struktur kelas final.

2. Penerapan Konsep OOP

2.1 Inheritance & Polymorphism

Kami mengimplementasikan *inheritance* dan *polymorphism* pada beberapa aspek program kita yaitu pada, kelas tool, kelas nontool, dan kelas exception. Kelas tool dan nontool meng-*inherit* kelas item yang meng-generalisasi kedua jenis item tersebut. Kami melakukan implementasi inheritance pada kelas tool dan nontool dengan tujuan agar item tool maupun nontool dapat diletakkan dalam slot dan diperlakukan sebagai kelas yang sama (*Polymorphism*) yaitu kelas Item. Sedangkan pada kelas exception, seluruh exception meng-*inherit* kelas BaseException untuk menggeneralisasi seluruh exception sehingga bisa dilakukan try-catch dengan menangkap seluruh jenis exception.

Berikut adalah cuplikan dan penjelasan kode kami yang kami implementasi

Kelas Tools dan Nontools yang meng-*inherit* kelas Item

```

1  class Tools : public Item
2  {
3  private:
4      int durability;
5
6  public:
7      Tools();
8      Tools(int, string, string, int);
9      Tools(int, string, int);
10     Tools(const Tools &);
11     string print() const;
12     int getQuantity() const;
13     int getDurability() const;
14     void decreaseDurability(int);
15     bool isTool();
16 };

```

```

1  class Nontools : public Item {
2  private:
3      int quantity;
4
5  public:
6      Nontools();
7      Nontools(int, string, string, int);
8      Nontools(int, string, int);
9      Nontools(const Nontools &);
10     string print() const;
11     int getQuantity() const;
12     void addQuantity(int);
13     bool isNontool();
14 };

```

Polymorphism pada file “Menu.cpp” di metode give, memperlakukan Nontools dan Tools sebagai kelas yang sama, yaitu kelas Item.

```

1 while (i < 27 && qty > 0) {
2     Item *s = getStorageElmtAtIdx(i);
3     if (s->getName() == "-") {
4         if (qty <= 64) {
5             setStorageAtIdx(
6                 i,
7                 new Nontools(items.getID(name), name, items.getType(name), qty),
8                 getStorageSlotName(i));
9             qty = 0;
10        } else {
11            setStorageAtIdx(
12                i, new Nontools(items.getID(name), name, items.getType(name), 64),
13                getStorageSlotName(i));
14            qty -= 64;
15        }
16    }
17    i++;
18 }

```

```

1 while (i < 27 && qty > 0) {
2     Item *s = getStorageElmtAtIdx(i);
3     // if the slot is empty add the item
4     if (s->getName() == "-") {
5         setStorageAtIdx(i, new Tools(items.getID(name), name, dura),
6                         getStorageSlotName(i));
7         qty--;
8     }
9     i++;
10 }

```

2.2 Method/Operator Overloading

Kami mengimplementasikan *Method/Operator Overloading* pada beberapa aspek program yaitu pada kelas Menu. Method overloading digunakan karena terdapat satu fungsi yang memiliki *behaviour* yang berbeda pada suatu fungsi yang sama. Untuk kasus kelas Menu ini jika kita ingin memberikan suatu tools baru dengan durability tertentu maka kita harus masukkan jumlah durability pada parameter fungsi, namun apa yang terjadi jika kita ingin memberikan nontools item? Maka int durability tidak akan terpakai karena nontool tidak memiliki durability. Maka digunakanlah Method overloading agar dapat menangani kasus ini.

Overloading method give pada Menu.cpp

give(ItemsReader&items,
string name, int qty, int dura)

Untuk memberikan item
berupa tools dengan durability
khusus


```

1 void Menu::give(ItemsReader &items, string name, int qty, int dura) {
2     if (items.getCtg(name) == "TOOL") {
3         /* Cari slot untuk dimasukkan tool */
4         int i = 0;
5         // loop through the inv
6         while (i < 27 && qty > 0) {
7             Item *s = getStorageElmtAtIdx(i);
8             // if the slot is empty add the item
9             if (s->getName() == "-") {
10                 setStorageAtIdx(i, new Tools(items.getID(name), name, dura),
11                     getStorageSlotName(i));
12                 qty--;
13             }
14             i++;
15         }
16     } else {
17         throw new ItemNotFoundException(name);
18     }
19 }

```


give(ItemsReader &items,
string name, int qty)

Memberikan item secara
umum



```
1 void Menu::give(ItemsReader &items, string name, int qty) {  
2     if (items.getCtg(name) == "TOOL") {  
3         give(items, name, qty, 10);  
4     } else if (items.getCtg(name) == "NONTOL") {  
5         int i = 0;  
6         while (i < 27 && qty > 0) {  
7             Item *s = getStorageElmtAtIdx(i);  
8             if (s->getName() == name) {  
9                 if (qty + s->getQuantity() <= 64) {  
10                     s->addQuantity(qty);  
11                     qty = 0;  
12                 } else {  
13                     int sisa = 64 - s->getQuantity();  
14                     s->addQuantity(sisa);  
15                     qty -= sisa;  
16                 }  
17             }  
18             i++;  
19         }  
20     }  
21 }
```

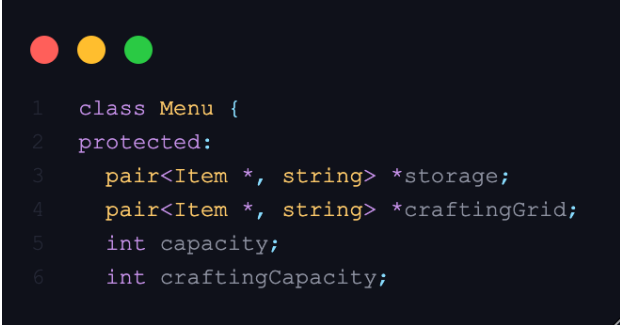
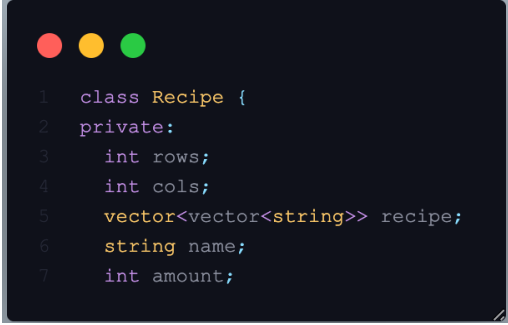
```

1  /* Cari slot yang kosong untuk dimasukkan nontool */
2      i = 0;
3      while (i < 27 && qty > 0) {
4          Item *s = getStorageElmtAtIdx(i);
5          if (s->getName() == "-") {
6              if (qty <= 64) {
7                  setStorageAtIdx(
8                      i,
9                      new Nontools(items.getID(name), name, items.getType(name), qty),
10                     getStorageSlotName(i));
11                  qty = 0;
12              } else {
13                  setStorageAtIdx(
14                      i, new Nontools(items.getID(name), name, items.getType(name), 64),
15                     getStorageSlotName(i));
16                  qty -= 64;
17              }
18          }
19          i++;
20      }
21      if (qty > 0) {
22          throw new InventoryFullException();
23      }
24  }
25  }

```

2.3 Template & Generic Classes


Kami juga mengimplementasikan generic pada pengerjaan Tugas Besar ini. Generic yang kami gunakan ialah generic yang sudah ada pada built in library C++ yaitu generic yang sering digunakan ketika menginstansiasi STL. Template & Generic berguna untuk mengurangi redundansi ketika membuat suatu class atau method agar tidak perlu melakukan method overloading berulang kali. Pada pengerjaan Tugas Besar ini terdapat beberapa Generic yang digunakan diantaranya

Generic yang digunakan	Screenshot
Item* & string pada pembuatan inventory	 <pre> 1 class Menu { 2 protected: 3 pair<Item *, string> *storage; 4 pair<Item *, string> *craftingGrid; 5 int capacity; 6 int craftingCapacity; </pre>
vector<string> pada pembuatan resep	 <pre> 1 class Recipe { 2 private: 3 int rows; 4 int cols; 5 vector<vector<string>> recipe; 6 string name; 7 int amount; </pre>

Recipe pada pembacaan resep



```
1 class RecipesReader {  
2 private:  
3     vector<Recipe> recipes;
```

String pada pemindahan item

```
1  if (command == "MOVE") {
2      string slot;
3      int N;
4      cin >> slot >> N;
5      if (slot[0] == 'I') {
6          vector<string> dest;
7          for (int i = 0; i < N; i++) {
8              string desSlot;
9              cin >> desSlot;
10             dest.push_back(desSlot);
11         }
12         if (dest[0][0] == 'I') {
13             this->menu->MoveInventory(slot, dest[0]);
14         } else if (dest[0][0] == 'C') {
15             /* Pindah dari inventory ke inventory */
16             this->menu->MoveToCraft(slot, N, dest);
17         } else {
18             throw new InvalidSlotIDException(dest[0]);
19         }
20     } else if (slot[0] == 'C') {
21         /* Pindahkan crafting ke inventory */
22         string dest;
23         cin >> dest;
24         this->menu->MoveFromCraft(slot, dest);
25     }
```

2.4 Exception

Kami mengimplementasikan *Exception* pada seluruh bagian kode kami agar dapat menampilkan error sesuai dengan kejadian diluar ekspektasi program. Dengan menggunakan *Exception* yang di-*throw*, maka tidak perlu mempertimbangkan kedalaman kode sehingga bebas melakukan *throw* dalam kode, asal di dalam blok *try-catch* yang membungkus seluruh program. *Exception* ini juga meng-implementasi konsep *inheritance* (lebih tepatnya Abstract Base Class) karena seluruh kelas *Exception* meng-*inherit* kelas *BaseException* yang membuat lebih mudah untuk ditangkap.

Berikut adalah cuplikan dan penjelasan kode kami yang kami implementasi

Implementasi dari Abstract Base Class "BaseException"

```
1 // Base Exception Class (Pure Virtual)
2 class BaseException {
3 public:
4     virtual void what() = 0;
5 };
```

Beberapa contoh Exception yang diimplementasi

```
1 // Item
2 class ItemNotFoundException : public BaseException {
3 private:
4     string item;
5
6 public:
7     ItemNotFoundException(string item) { this->item = item; }
8     void what() { cout << this->item << " not found!" << endl; }
9 };
10
11 class WrongItemTypeException : public BaseException {
12 private:
13     Item *item;
14
15 public:
16     WrongItemTypeException(Item *item) { this->item = item; }
17     void what() {
18         string rightType = this->item->isTool() ? "nontool" : "tool";
19         cout << "Item " << this->item->getName() << " is not a " << rightType << "!"
20             << endl;
21     }
22 };
```

Salah satu penggunaan Exception pada Menu

```
1 // check if the slot is available or not
2 int Menu::checkId(string Id, string arrayType) {
3     int i = 0;
4     i = stoi(Id.substr(1));
5     string header = Id.substr(0, 1);
6     if (arrayType == "INVENTORY" && header == "C") {
7         throw new InvalidSlotIDException(Id);
8     }
9     if (arrayType == "CRAFT" && header == "I") {
10        throw new InvalidSlotIDException(Id);
11    }
12
13    if (arrayType == "INVENTORY" && (i < 0 || i > 26)) {
14        throw new InvalidSlotIDException(Id);
15    } else if (arrayType == "CRAFT" && (i < 0 || i > 8)) {
16        throw new InvalidSlotIDException(Id);
17    }
18    return i;
19 }
```


Salah satu bagian yang menangkap Exception yang di 'throw'

```

1 void Game::StartGame() {
2     cout << "Welcome to the MineHati!\nType HELP to get started.\n\n";
3     while (!gameEnd) {
4         try {
5             string command = askCommand();
6             process(command);
7         } catch (BaseException *e) {
8             // Clear cin
9             cin.clear();
10            cin.ignore(numeric_limits<streamsize>::max(), '\n');
11
12            // Output Exception
13            e->what();
14            cout << endl;
15        }
16    }
17 }

```

2.5 C++ Standard Template Library

Kami memanfaatkan STL dari library C++ pada bagian reader dan menu untuk menyimpan data pada struktur data yang sudah disediakan sehingga meningkatkan keefisienan saat digunakan. Pada reader kami menggunakan map untuk menyimpan data item yang telah dibaca dari text file, sedangkan pada menu, kami menggunakan pair untuk menyimpan data item yang berada pada crafting maupun inventory. Dengan menggunakan STL ini, kita dapat memanfaatkan sifat dinamis dari array yang sulit diimplementasi pada bahasa C.

Berikut adalah cuplikan dan penjelasan kode kami yang kami implementasi

Penggunaan STL map pada class ItemsReader

```
1  class ItemsReader {
2  private:
3      map<string, int> idMap;
4      map<string, string> typeMap;
5      map<string, string> ctgMap;
6
7  public:
8      ItemsReader(string fileName) {
9          ifstream inFile;
10         inFile.open(fileName);
11         if (!inFile) {
12             throw new FileNotFoundException(fileName);
13         }
14         int id;
15         string name;
16         string type;
17         string ctg;
18         while (inFile >> id >> name >> type >> ctg) {
19             this->idMap[name] = id;
20             this->typeMap[name] = type;
21             this->ctgMap[name] = ctg;
22         }
23         inFile.close();
24     }
25     int getID(string name) { return this->idMap[name]; }
26     string getType(string name) { return this->typeMap[name]; }
27     string getCtg(string name) { return this->ctgMap[name]; }
28 };
```

Penggunaan STL vector pada class Recipe

```
1  class Recipe {
2  private:
3      int rows;
4      int cols;
5      vector<vector<string>> recipe;
6      string name;
7      int amount;
8
9  public:
10     Recipe(int rows, int cols, vector<vector<string>> recipe, string name,
11            int amount) {
12         this->rows = rows;
13         this->cols = cols;
14         this->recipe = recipe;
15         this->name = name;
16         this->amount = amount;
17     }
18     string getName() { return this->name; }
19     int getAmount() { return this->amount; }
20     int getRows() { return this->rows; }
21     int getCols() { return this->cols; }
22     vector<vector<string>> getRecipe() { return this->recipe; }
23 };
```

Penggunaan STL pair pada class Menu

```
1 class Menu {  
2     protected:  
3         pair<Item *, string> *storage;  
4         pair<Item *, string> *craftingGrid;  
5         int capacity;  
6         int craftingCapacity;  
7 }
```

2.6 Abstract Base Class (Konsep OOP Lain)

Kami menggunakan Abstract Base Class dalam implmentasi Exception. Seluruh Exception meng-*inherit* suatu Abstract Base Class yaitu kelas BaseException yang hanya berisikan metode pure virtual. Abstract Base Class berguna sebagai blueprint dari kelas turunannya agar berbentuk sesuai dengan sutau template.

Berikut adalah cuplikan dan penjelasan kode kami yang kami implementasi

Abstract Base Class BaseException

```
1 // Base Exception Class (Pure Virtual)
2 class BaseException {
3 public:
4     virtual void what() = 0;
5 };
```

Salah satu kelas yang mengikuti blueprint dari BaseException

```
1 // IO Failed Exception Class
2 class IOException : public BaseException {
3 private:
4     BaseException *exc;
5
6 public:
7     void what() {
8         cout << "IO Failed!" << endl;
9         exc->what();
10    }
11 };
```

3. Bonus Yang dikerjakan

3.1 Bonus yang diusulkan oleh spek

3.1.1 Multiple Crafting

Ada satu perubahan di bagian *crafting* yaitu satu *slot crafting* sekarang bisa menampung lebih banyak dari satu *item* agar bisa melakukan *multiple crafting*. Berikut adalah contoh jalannya *multiple crafting* pada permainan MineHati.

Dibuat 8 STICK dalam satu kali *crafting*.

```
Masukkan command: SHOW
[C0 EMPTY] [C1 EMPTY] [C2 EMPTY]
[C3 EMPTY] [C4 OAK_PLANK 3] [C5 EMPTY]
[C6 EMPTY] [C7 OAK_PLANK 2] [C8 EMPTY]

[I0 EMPTY] [I1 EMPTY] [I2 EMPTY] [I3 EMPTY] [I4 EMPTY] [I5 EMPTY] [I6 EMPTY] [I7 EMPTY] [I8 EMPTY]
[I9 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]

Masukkan command: CRAFT
Masukkan command: SHOW
[C0 EMPTY] [C1 EMPTY] [C2 EMPTY]
[C3 EMPTY] [C4 OAK_PLANK 1] [C5 EMPTY]
[C6 EMPTY] [C7 EMPTY] [C8 EMPTY]

[I0 STICK 8] [I1 EMPTY] [I2 EMPTY] [I3 EMPTY] [I4 EMPTY] [I5 EMPTY] [I6 EMPTY] [I7 EMPTY] [I8 EMPTY]
[I9 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]
```

Dibuat WOODEN_PICKAXE dan WOODEN_SHOVEL dalam satu kali *crafting*.

```
Masukkan command: SHOW
[C0 OAK_PLANK 1] [C1 OAK_PLANK 2] [C2 OAK_PLANK 1]
[C3 EMPTY] [C4 STICK 2] [C5 EMPTY]
[C6 EMPTY] [C7 STICK 2] [C8 EMPTY]

[I0 EMPTY] [I1 EMPTY] [I2 EMPTY] [I3 EMPTY] [I4 EMPTY] [I5 EMPTY] [I6 EMPTY] [I7 EMPTY] [I8 EMPTY]
[I9 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]

Masukkan command: CRAFT
Masukkan command: SHOW
[C0 EMPTY] [C1 EMPTY] [C2 EMPTY]
[C3 EMPTY] [C4 EMPTY] [C5 EMPTY]
[C6 EMPTY] [C7 EMPTY] [C8 EMPTY]

[I0 WOODEN_PICKAXE [10/10]] [I1 WOODEN_SHOVEL [10/10]] [I2 EMPTY] [I3 EMPTY] [I4 EMPTY] [I5 EMPTY] [I6 EMPTY] [I7 EMPTY] [I8 EMPTY]
[I9 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]
```

Multiple crafting diimplementasikan dengan menambahkan iterasi pada implementasi awal kode *CRAFT* supaya *crafting* diulangi sampai tidak ditemukan resep lagi (implementasi sebelumnya langsung memberhentikan *craft* ketika sudah menemukan 1 resep saja).

Implementasi *multiple crafting* dilakukan dengan cara mengasumsikan di kondisi awal bahwa *recipeFound* = *true*. *Crafting* dilakukan terus menerus sampai *recipeFound* = *false*.

```

1  while (recipeFound) {
2      recipeFound = false;
3      /* Cek setiap resep yang ada di recipe */
4      for (int x = 0; x < r.size() && !recipeFound; x++) {
5          vector<vector<string>> recipe = r[x].getRecipe();
6          int rows = r[x].getRows();
7          int cols = r[x].getCols();
8          if (rows != this->getCraftingRows() || cols != this->getCraftingCols()) {
9              continue;
10         }
11         /* Looping pada matrix c */
12         bool foundDifferent = false;
13         for (int i = 0; i < 3 - rows + 1 && !recipeFound; i++) {
14             for (int j = 0; j < 3 - cols + 1 && !recipeFound; j++) {
15                 foundDifferent = false;
16                 for (int k = 0; k < rows && !foundDifferent; k++) {
17                     for (int l = 0; l < cols && !foundDifferent; l++) {
18                         if (this->getElement(i + k, j + l).first->getName() != "-") {
19                             string type = items.getType(
20                                 this->getElement(i + k, j + l).first->getName());
21                             if (type != "-") {
22                                 /* Ada tipenya */
23                                 foundDifferent = type != recipe[k][l];
24                             } else {
25                                 /* Tidak ada tipenya */
26                                 foundDifferent =
27                                     this->getElement(i + k, j + l).first->getName() !=
28                                     recipe[k][l];
29                             }
30                         } else {
31                             foundDifferent =
32                                 this->getElement(i + k, j + l).first->getName() !=
33                                 recipe[k][l];
34                         }
35                     }
36                 }
37                 /* k dan l out of range atau foundDifferent = true */
38                 if (!foundDifferent) {
39                     recipeFound = true;
40                 }
41             }
42         }
43
44         if (recipeFound) {
45             ++ recipeCount;
46             this->decreaseCraftingByOne();
47             give(items, r[x].getName(), r[x].getAmount());
48         }
49     }
50 }
51 }

```

3.1.2 Item dan Tool Baru

Tool baru yang kami tambah ke dalam permainan MineHati, yaitu::

- WOODEN_SHOVEL
- STONE_SHOVEL
- IRON_SHOVEL
- DIAMOND_SHOVEL
- WOODEN_HOE
- STONE_HOE
- IRON_HOE
- DIAMOND_HOE

Nontool baru yang kami tambah ke dalam permainan MineHati, yaitu:

- COAL
- TORCH
- WORKBENCH
- CHEST
- FURNACE

3.1.3 Unit Testing Implementation

Implementasi unit testing dilakukan dalam file-file berakhiran *Test.cpp* pada masing-masing folder modul. Masing-masing file .cpp tersebut adalah sebuah *driver* untuk menjalankan setiap modul yang ada. Kode program-program tersebut ditampilkan pada tabel di bawah ini.


Modul Exception	 <pre>1 #include "Exception.hpp" 2 #include <string> 3 4 const string exceptionToTest = "FileNotFoundException"; 5 6 int main() { 7 try { 8 if (exceptionToTest == "InvalidCommandException") { 9 throw new InvalidCommandException("EXCEPTION_TEST"); 10 } 11 if (exceptionToTest == "FileNotFoundException") { 12 throw new FileNotFoundException("EXCEPTION_TEST"); 13 } 14 cout << "Test ran without finding any exceptions!" << endl; 15 } catch (BaseException* e) { 16 e->what(); 17 } 18 return 0; 19 }</pre>
-----------------	--

Modul ItemsReader	 <pre>1 #include "ItemsReader.hpp" 2 3 int main() { 4 ItemsReader items("../config/item.txt"); 5 cout << items.getID("OAK_PLANK") << endl; 6 cout << items.getCtg("OAK_PLANK") << endl; 7 cout << items.getType("OAK_PLANK") << endl; 8 }</pre>
Modul RecipesReader	 <pre>1 #include "RecipesReader.hpp" 2 3 int main() { 4 RecipesReader r("../config/recipe"); 5 vector<Recipe> recipes = r.getRecipes(); 6 for (Recipe recipe : recipes) { 7 cout << "===== " << endl; 8 cout << recipe.getName() << endl; 9 cout << recipe.getAmount() << endl; 10 cout << recipe.getRows() << endl; 11 cout << recipe.getCols() << endl; 12 for (int i = 0; i < recipe.getRows(); i++) { 13 for (int j = 0; j < recipe.getCols(); j++) { 14 cout << recipe.getRecipe()[i][j] << " "; 15 } 16 cout << endl; 17 } 18 } 19 }</pre>

Modul Item


```
1  #include "item.hpp"
2  #include <iostream>
3  using namespace std;
4
5  void printItem(Item& i) {
6      cout << "=====" << endl;
7      cout << i.getId() << endl;
8      cout << i.getName() << endl;
9      cout << i.getCategory() << endl;
10 }
11
12 int main() {
13     Item defaultItem;
14     Item userDefinedItem(3, "BIRCH_PLANK", "PLANK");
15     printItem(defaultItem);
16     printItem(userDefinedItem);
17 }
```

Modul Nontools



```
1 #include "nontools.hpp"
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     Nontools defaultNontools;
7     cout << defaultNontools.print() << endl;
8     Nontools userDefinedNontools(3, "BIRCH_PLANK", 3);
9     cout << userDefinedNontools.print() << endl;
10    userDefinedNontools.addQuantity(20);
11    if (userDefinedNontools.isNontool()) {
12        cout << userDefinedNontools.print() << endl;
13    }
14 }
```

Modul Tools




```
1  #include "tools.hpp"
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6      Tools defaultTools;
7      cout << defaultTools.print() << endl;
8      Tools userDefinedNontools(3, "WOODEN_SWORD", 10);
9      cout << userDefinedNontools.print() << endl;
10     userDefinedNontools.decreaseDurability(5);
11     if (userDefinedNontools.isTool()) {
12         cout << userDefinedNontools.print() << endl;
13     }
14 }
```

Modul Menu



```
1  #include "Menu.hpp"
2
3  int main() {
4      Menu menu;
5      ItemsReader items = ItemsReader("../config/item.txt");
6      RecipesReader recipes = RecipesReader("../config/recipe");
7      menu.Show();
8      menu.give(items, "OAK_PLANK", 1);
9      menu.give(items, "BIRCH_PLANK", 2);
10     menu.give(items, "STICK", 1);
11     menu.MoveToCraft("I0", 1, {"C0"});
12     menu.MoveToCraft("I1", 1, {"C3"});
13     menu.Craft(items, recipes);
14     menu.Discard("I2", 2);
15     menu.Show();
16     menu.give(items, "WOODEN_SWORD", 1);
17     menu.Use("I0");
18     menu.Show();
19     menu.exportInventory(items, "./menuTest.out");
20 }
```

Modul Game



```
1  #include "../game.hpp"
2  using namespace std;
3
4  int main(){
5      cout << "1. Filepath is wrong\n";
6      //
7      try {
8          Game game;
9      } catch (BaseException *e) {
10         e->what();
11     }
12     cout << "2. Filepath is correct\n";
13     string dir = "../../../config";
14     Game game(dir);
15
16     cout << "3. Command is wrong\n";
17     try {
18         game.process("AFQWF");
19     } catch (BaseException *e){
20         e->what();
21     }
22
23     cout << "4. Command is correct\n";
24     game.process("SHOW");
25
26     return 0;
27 }
```

3.2 Bonus Kreasi Mandiri

3.2.1 *Command* RECIPES

Command RECIPES menunjukkan resep apa saja yang bisa dibuat, lengkap dengan bahan-bahannya. Berikut adalah hasil dari penggunaan *command* RECIPES.

```
Masukkan command: RECIPES
CHEST: 1
PLANK    PLANK    PLANK
PLANK    -        PLANK
PLANK    PLANK    PLANK

DIAMOND_AXE: 1
DIAMOND  DIAMOND
DIAMOND  STICK
-        STICK

DIAMOND_HOE: 1
DIAMOND  DIAMOND
-        STICK
-        STICK

DIAMOND_PICKAXE: 1
DIAMOND  DIAMOND DIAMOND
-        STICK  -
-        STICK  -

DIAMOND_SHOVEL: 1
DIAMOND
STICK
STICK
```



```

DIAMOND_SWORD: 1
DIAMOND
DIAMOND
STICK

FURNACE: 1
STONE  STONE  STONE
STONE  -      STONE
STONE  STONE  STONE

IRON_AXE: 1
IRON_INGOT      IRON_INGOT
IRON_INGOT      STICK
-              STICK

IRON_HOE: 1
IRON_INGOT      IRON_INGOT
-              STICK
-              STICK

IRON_INGOT: 1
IRON_NUGGET      IRON_NUGGET      IRON_NUGGET
IRON_NUGGET      IRON_NUGGET      IRON_NUGGET
IRON_NUGGET      IRON_NUGGET      IRON_NUGGET

IRON_NUGGET: 9
IRON_INGOT

IRON_PICKAXE: 1
IRON_INGOT      IRON_INGOT      IRON_INGOT
-              STICK      -

```

```
-      STICK      -  
  
IRON_SHOVEL: 1  
IRON_INGOT  
STICK  
STICK  
  
IRON_SWORD: 1  
IRON_INGOT  
IRON_INGOT  
STICK  
  
OAK_PLANK: 4  
OAK_LOG  
  
SPRUCE_PLANK: 4  
SPRUCE_LOG  
  
STICK: 4  
PLANK  
PLANK  
  
STONE_AXE: 1  
STONE      STONE  
STONE      STICK  
-          STICK  
  
STONE_HOE: 1  
STONE      STONE  
-          STICK  
-          STICK
```

```
STONE_PICKAXE: 1
STONE    STONE    STONE
-        STICK    -
-        STICK    -

STONE_SHOVEL: 1
STONE
STICK
STICK

STONE_SWORD: 1
STONE
STONE
STICK

TORCH: 4
COAL
STICK

WOODEN_AXE: 1
PLANK    PLANK
PLANK    STICK
-        STICK

WOODEN_HOE: 1
PLANK    PLANK
-        STICK
-        STICK

WOODEN_PICKAXE: 1
PLANK    PLANK    PLANK
-        STICK    -
```

- STICK -

WOODEN_SHOVEL: 1

PLANK

STICK

STICK

WOODEN_SWORD: 1

PLANK

PLANK

STICK

WORKBENCH: 1

PLANK PLANK

PLANK PLANK

3.2.2 Command HELP

Command HELP memberikan informasi bagi pemain mengenai *command-command* apa saja yang dapat dieksekusi oleh pemain. Berikut adalah hasil eksekusi dari *command* HELP.

```
Masukkan command: HELP

Available commands:
SHOW: Show inventory and craft
DISCARD <INVENTORY_SLOT_ID> <QUANTITY>: Throw item in inventory with some
quantity
USE <slot>: Use tool in inventory
GIVE <ITEM_NAME> <QUANTITY>: Give specific item to with some quantity
MOVE <INVENTORY_SLOT_ID> N <CRAFT_SLOT_ID> / <INVENTORY_SLOT_ID> ...: Move item
in craft and inventory
CRAFT: Use craft to make new item that available in the recipe
EXPORT <FILENAME>: Export inventory and craft into a file
RECIPES: See all available recipe
HELP: To see list of commands
QUIT: To exit the game
```

3.2.3 Setting Directory Konfigurasi File

Pemain dapat melakukan konfigurasi secara mudah untuk menentukan letak disimpannya file konfigurasi (yaitu list item dan list recipe yang ada). Hal ini dapat dilakukan dengan mengubah configPath di file main.cpp lalu melakukan kompilasi ulang.



```
1  include "lib/Game/game.hpp"
2
3  int main()
4  {
5      try {
6          string configPath = "./config";
7          Game game(configPath);
8          game.StartGame();
9      } catch (BaseException *e) {
10         e->what();
11     }
12     return 0;
13 }
```

3.2.4 ASCII Art

Pada awal permainan, untuk memperindah interface yang diberikan untuk pemain, dibuat ASCII art yang menampilkan nama permainan yang kami buat, yaitu MineHati.

```
PS C:\Users\acoxs\Documents\GitHub\mineHati> ./main
Loaded with config located in ./config

/ $$      / $$ /$$$$$$$ / $$      / $$ /$$$$$$$$$ / $$      / $$ /$$$$$$$ /$$$$$$$$$ /$$$$$$$
| $$$     /$$$|_  $$_/ | $$$ | $$| $$_____/ | $$ | $$ /$$_  $$|_  $$_____/ |  $$$/
| $$$$ /$$$ | $$ | $$$$| $$| $$      | $$ | $$| $$ \ $$ | $$ | $$
| $$ $$/$$ $$ | $$ | $$ $$ $$| $$$$ | $$$$$$$$| $$$$$$$$ | $$ | $$
| $$ $$$ | $$ | $$ | $$ $$$$| $$_/ | $$_  $$| $$_  $$ | $$ | $$
| $$ \ / | $$ /$$$$$$$| $$ \  $$| $$$$$$$$| $$ | $$| $$ | $$ | $$ /$$$$$$$
|_/_/    |_/_/|_____/|_/_/ \/_/|_____/|_/_/ |_/_/|_/_/ |_/_/ |_/_/

Welcome to the MineHati!
Type HELP to get started.
```

4. Pembagian Tugas

Modul (dalam poin spek)	Implementer	Tester
I/O, Multiple Crafting, Unit Test	13520119	13520029, 13520062, 13520068, 13520101, 13520119
Menu, Inventory, Game	13520101	13520029, 13520062, 13520068, 13520101, 13520119
Items, Game	13520029	13520029, 13520062, 13520068, 13520101, 13520119
Menu, Craft	13520068	13520029, 13520062, 13520068, 13520101, 13520119
Exception, Integrate, Coverage	13520062	13520029, 13520062, 13520068, 13520101, 13520119