

Kode Kelompok : O08

Nama Kelompok : \* ~ ' . \* ♥ OOPin 1pin ♥ \* . ' ~ \*

1. 13521095 / Muhamad Aji Wibisono
2. 13521129 / Chiquita Ahsanunnisa
3. 13521149 / Rava Maulana Azzikri
4. 13521152 / Vanessa Rebecca Wiyono
5. 13521171 / Alisha Listya Wardhani

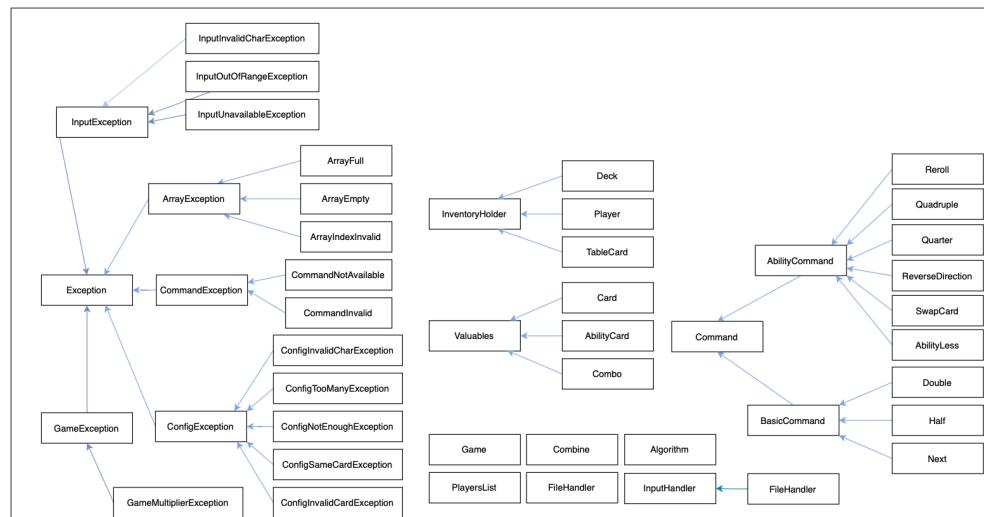
Asisten Pembimbing : Widya Anugrah Putra

Link repository : <https://github.com/ashnchiquita/Tubes-1-OOP-O08.git>

## 1. Diagram Kelas

### 1.1. Hirarki Kelas

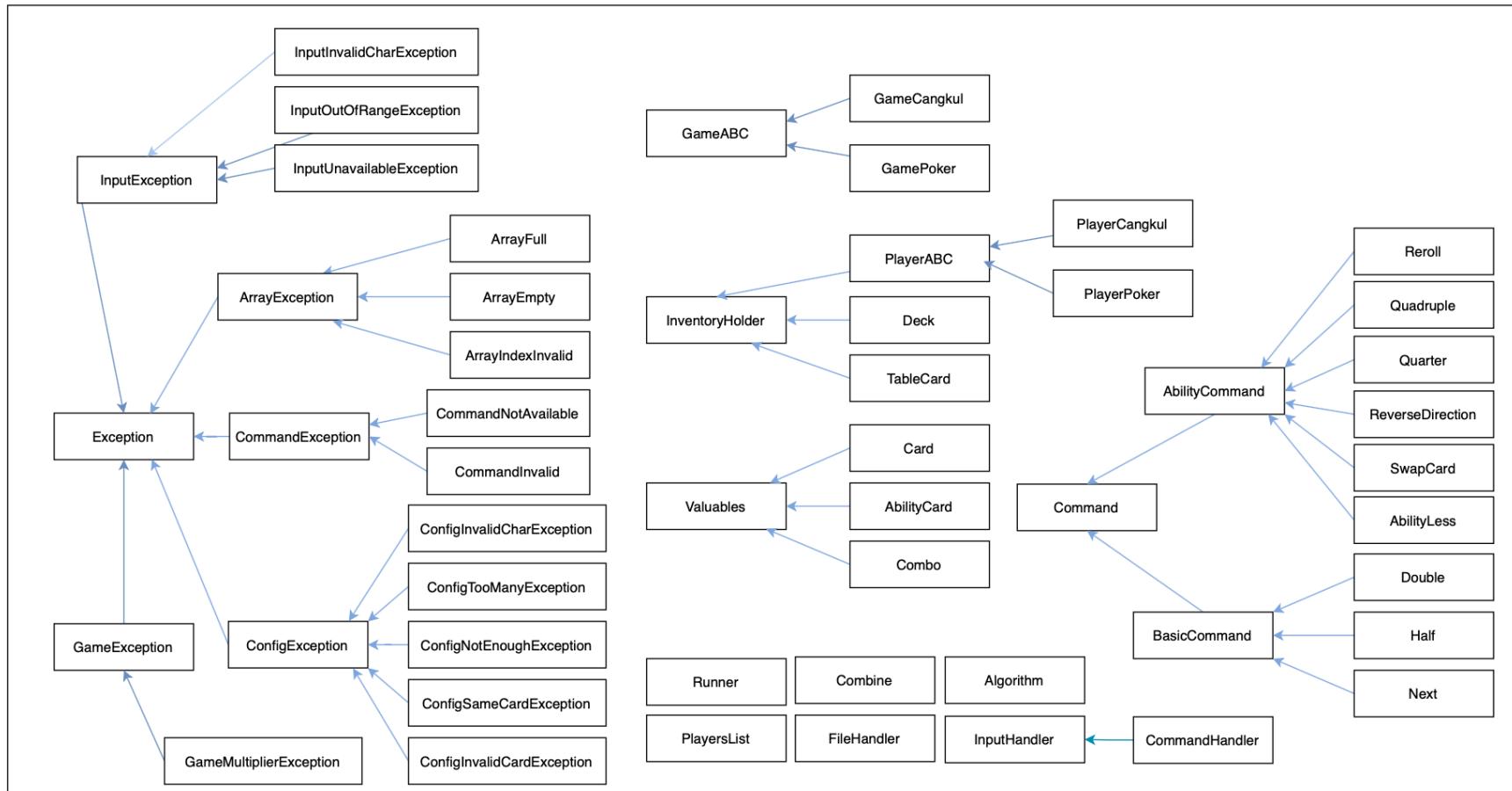
#### 1.1.1. Rancangan Tanpa Bonus



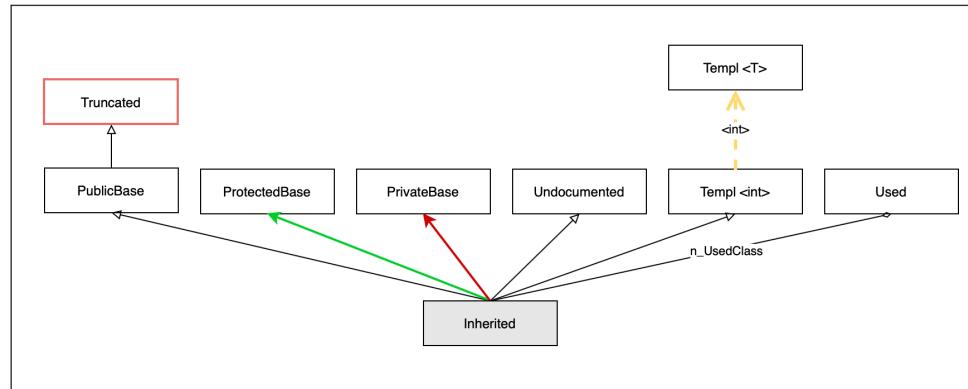
Berikut merupakan hierarki kelas pada program. Panah biru menunjukkan hubungan *inheritance*.

### 1.1.1. Rancangan dengan Bonus

Berikut merupakan rancangan hierarki kelas dengan bonus.



## 1.2. Diagram Kelas



### Legenda

Simpul pada graf tersebut mempunyai keterangan sebagai berikut:

- Kotak berwarna abu-abu merepresentasikan kelas yang diacu
- Kotak dengan *outline* hitam merepresentasikan kelas yang terdokumentasi
- Kotak dengan *outline* abu-abu merepresentasikan kelas yang tidak terdokumentasi
- Kotak dengan *outline* merah merepresentasikan kelas dimana tidak semua *inheritance* atau relasi *containment*-nya diperlihatkan.

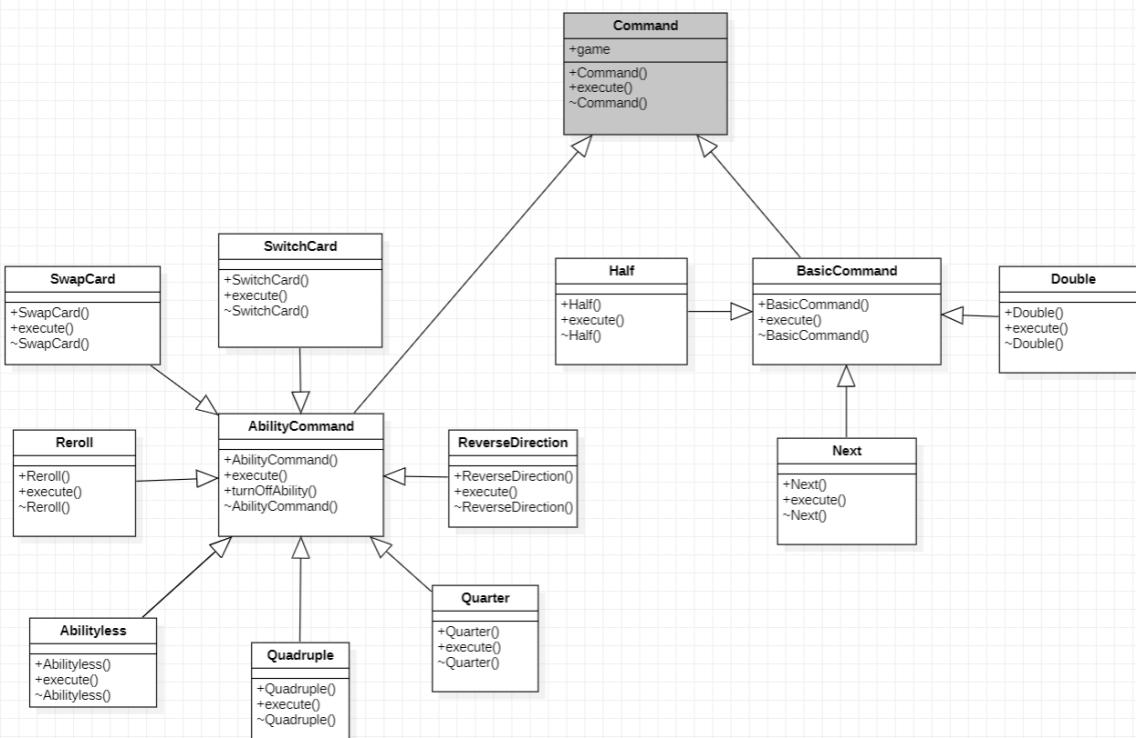
Panah pada graf tersebut mempunyai keterangan sebagai berikut:

- Panah hitam memvisualisasikan hubungan *public inheritance*
- Panah hijau memvisualisasikan hubungan *protected inheritance*
- Panah merah memvisualisasikan hubungan *private inheritance*
- Panah diamond memvisualisasikan jika sebuah kelas berada di dalam atau digunakan dalam suatu kelas. Panah dilabeli variabel atau atribut yang bisa diakses oleh kelas yang ditunjuk
- Panah kuning putus-putus memvisualisasikan relasi antara sebuah *template class* dan *template instance*.

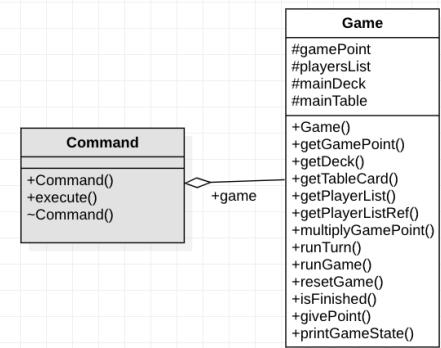
Berikut merupakan diagram dari kelas yang sudah diimplementasikan pada program.

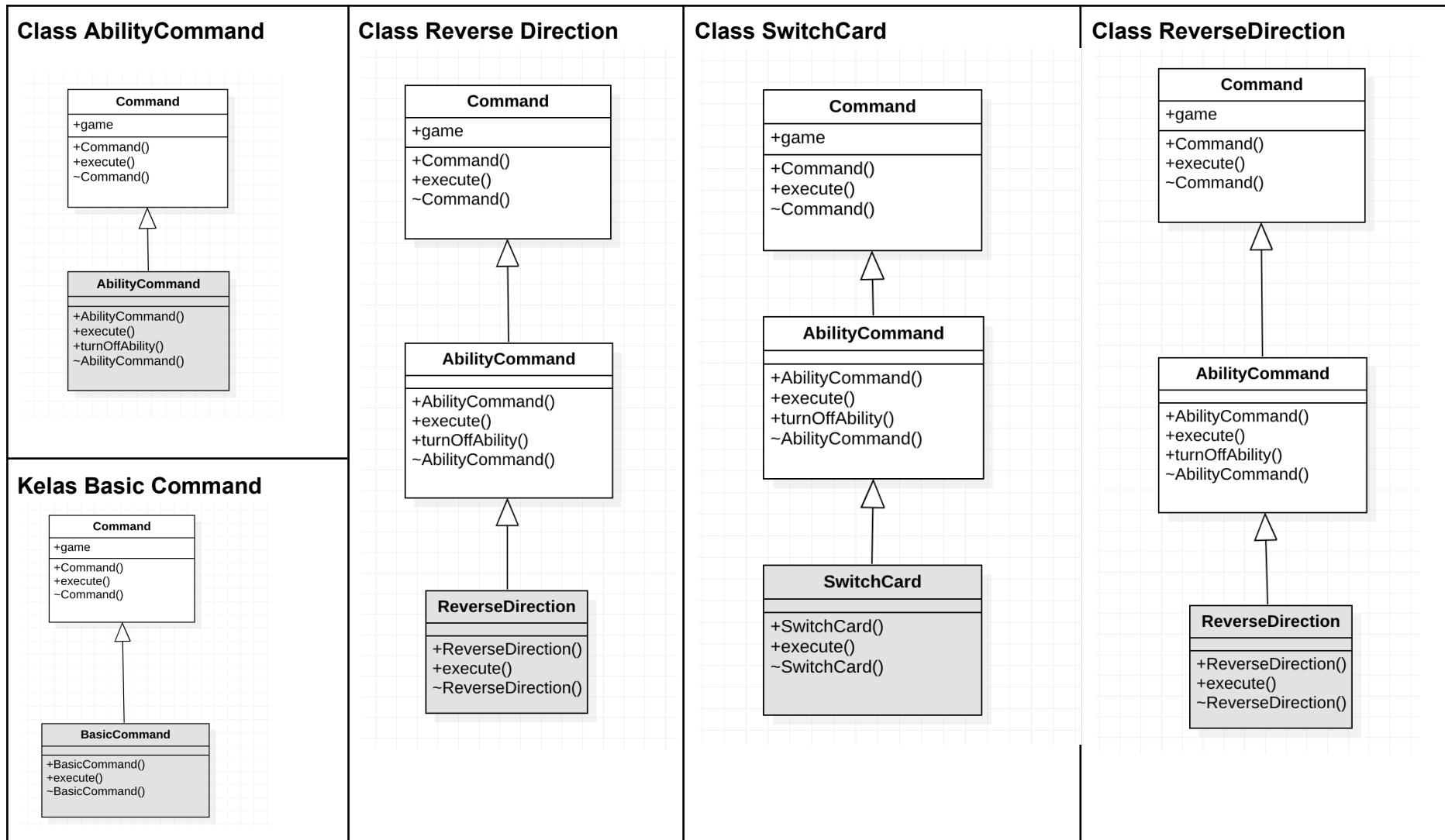
## Class Command

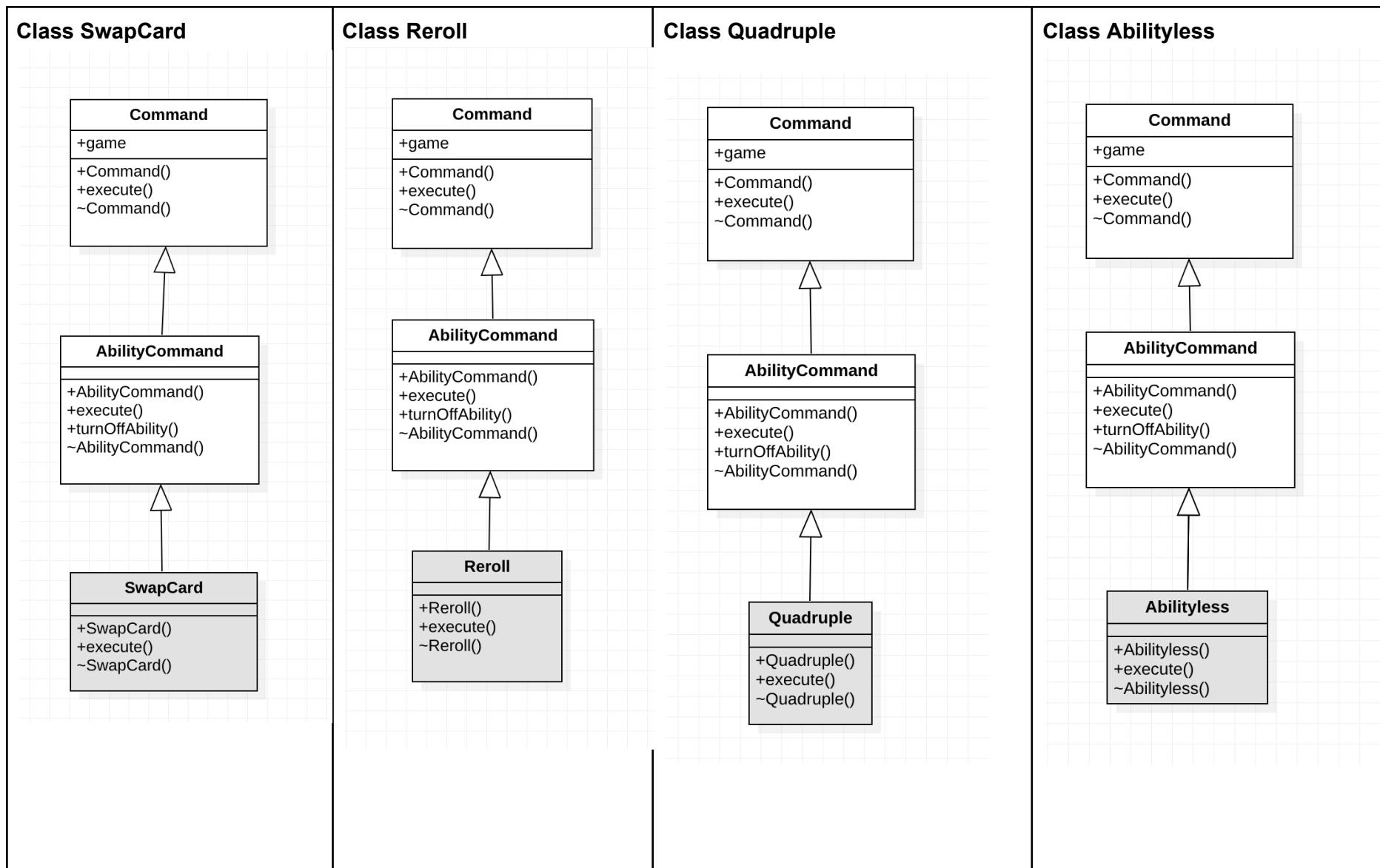
**Inheritance Diagram**  
**Class Command**

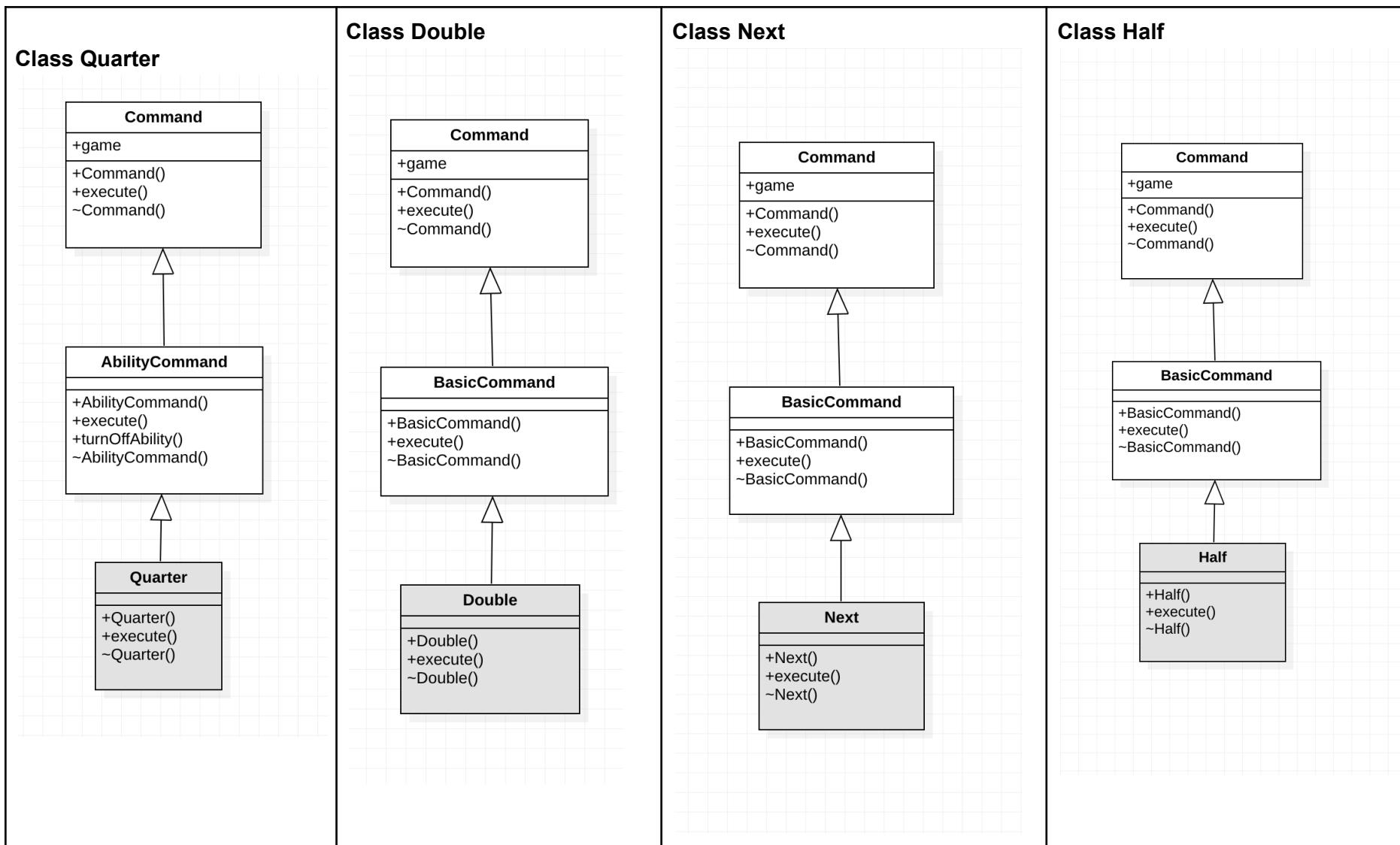


**Collaboration Diagram**  
**Class Command**



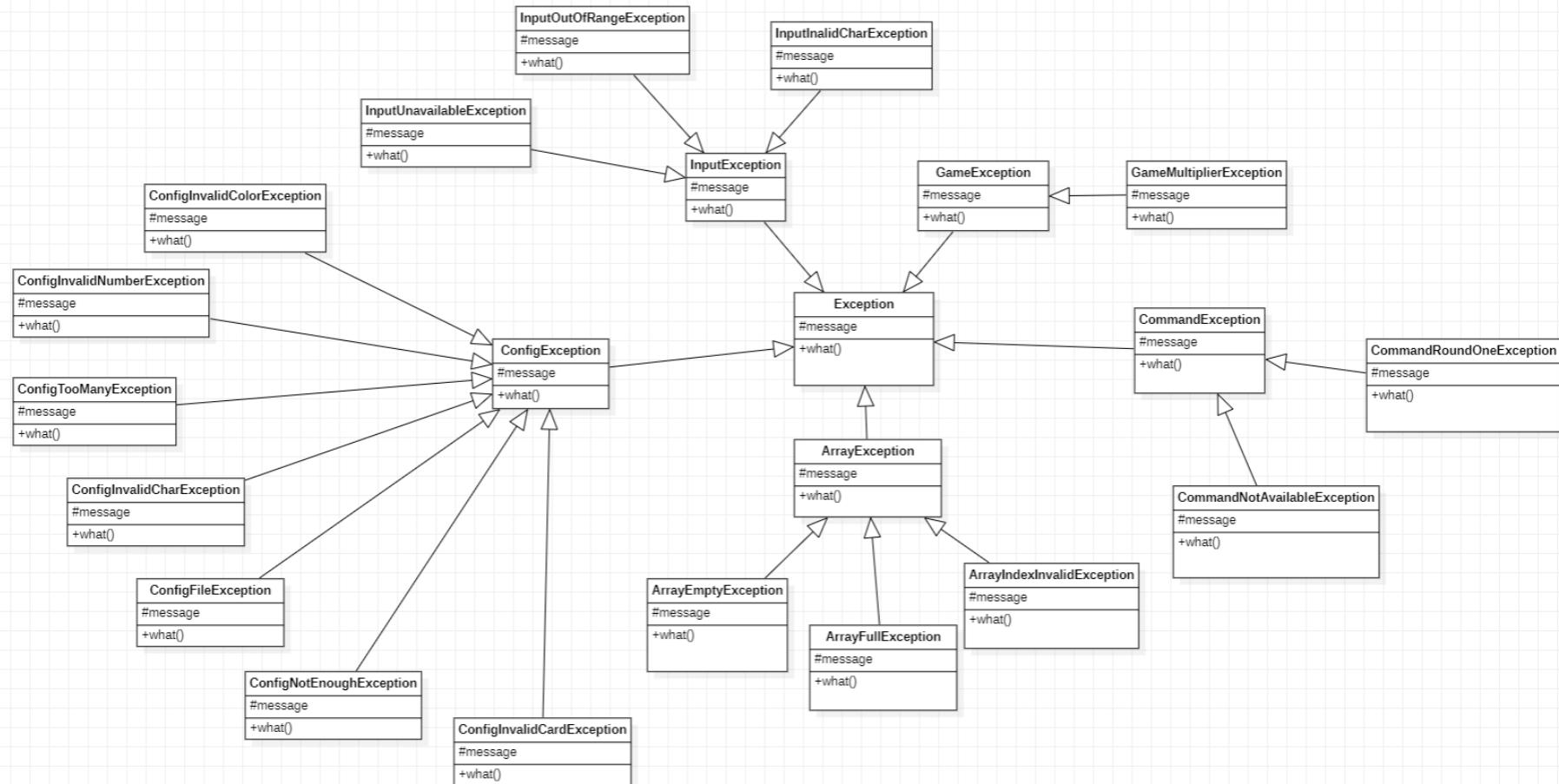






## Inheritance Diagram

### Class Exception

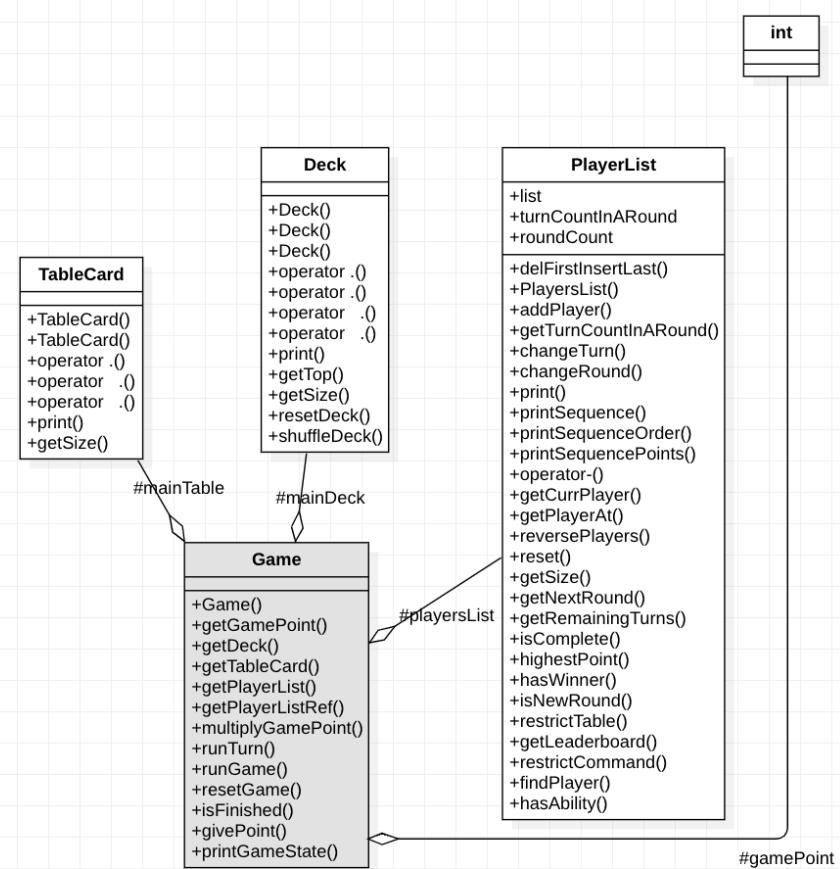


# Inheritance Diagram Class Game

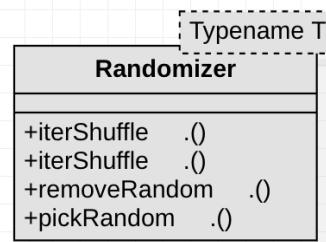


# Collaboration Diagram

## Class Game



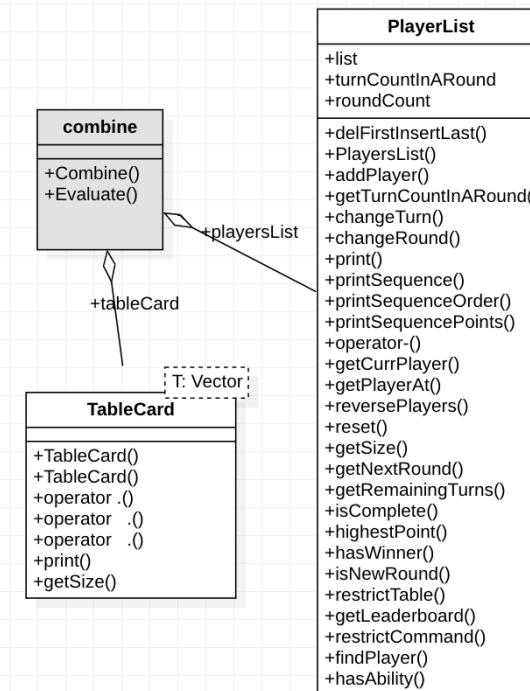
**Inheritance Diagram**  
**Class Randomizer**



**Inheritance Diagram**  
**Class Combine**



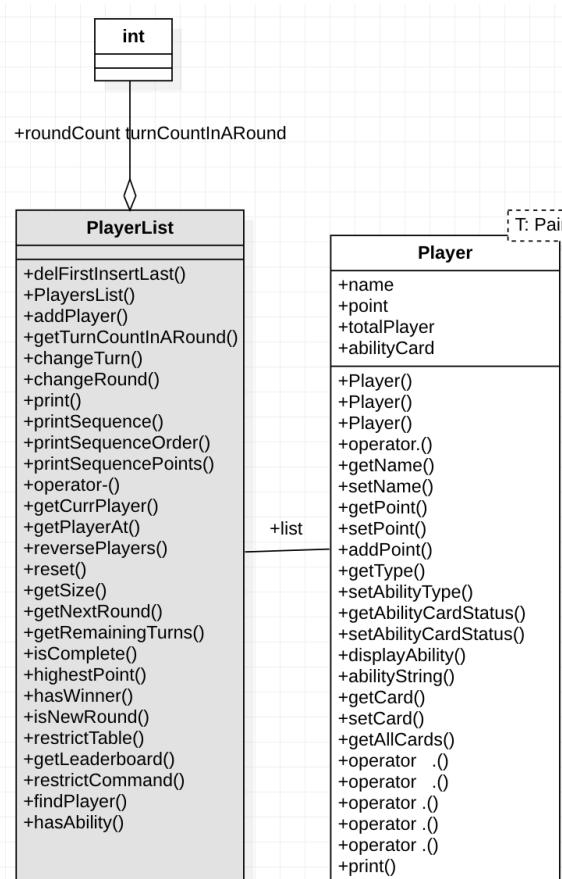
**Collaboration Diagram**  
**Class Combine**



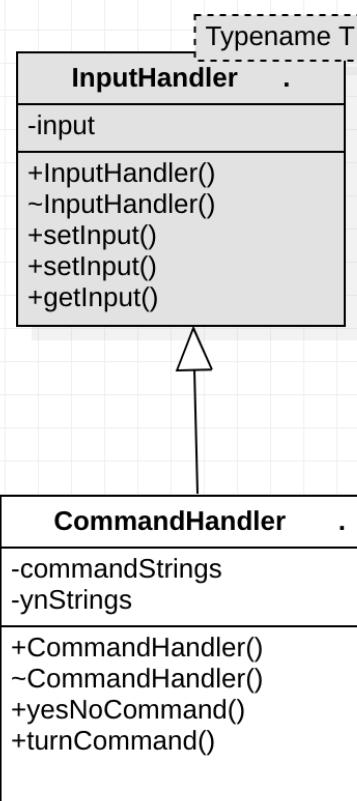
### Inheritance Diagram Class PlayersList

| PlayerList              |
|-------------------------|
| +list                   |
| +turnCountInARound      |
| +roundCount             |
| +delFirstInsertLast()   |
| +PlayersList()          |
| +addPlayer()            |
| +getTurnCountInARound() |
| +changeTurn()           |
| +changeRound()          |
| +print()                |
| +printSequence()        |
| +printSequenceOrder()   |
| +printSequencePoints()  |
| +operator -()           |
| +getCurrPlayer()        |
| +getPlayerAt()          |
| +reversePlayers()       |
| +reset()                |
| +getSize()              |
| +getNextRound()         |
| +getRemainingTurns()    |
| +isComplete()           |
| +highestPoint()         |
| +hasWinner()            |
| +isNewRound()           |
| +restrictTable()        |
| +getLeaderboard()       |
| +restrictCommand()      |
| +findPlayer()           |
| +hasAbility()           |

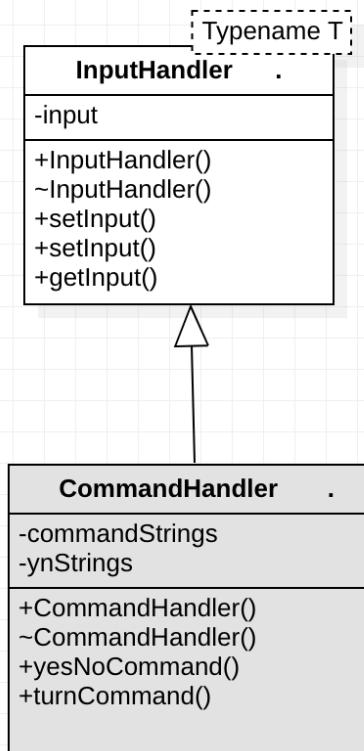
### Collaboration Diagram Class PlayersList



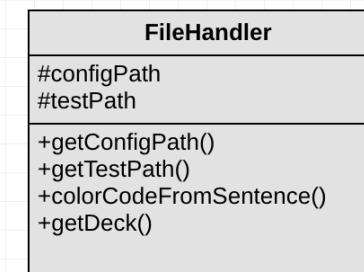
Inheritance Diagram  
Kelas InputHandler



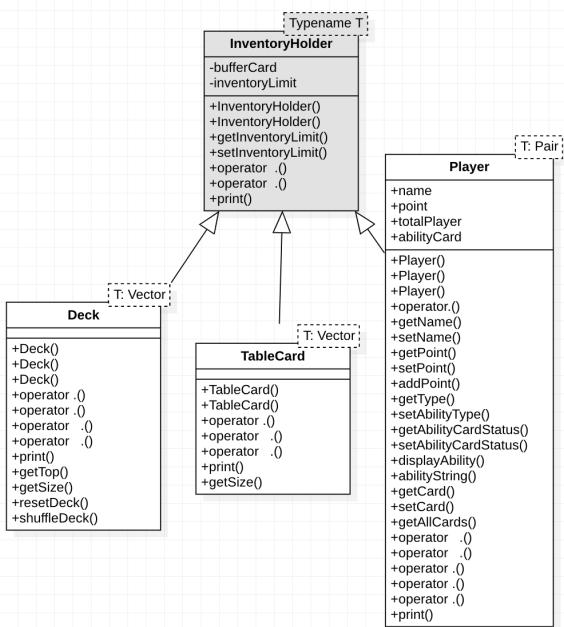
Inheritance Diagram  
Kelas CommandHandler



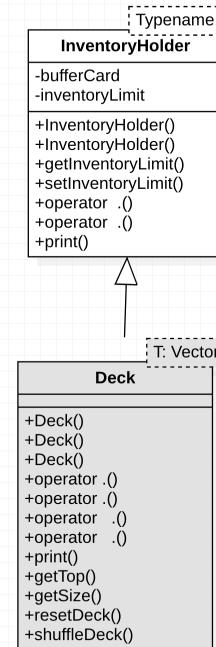
Inheritance Diagram  
Kelas FileHandler



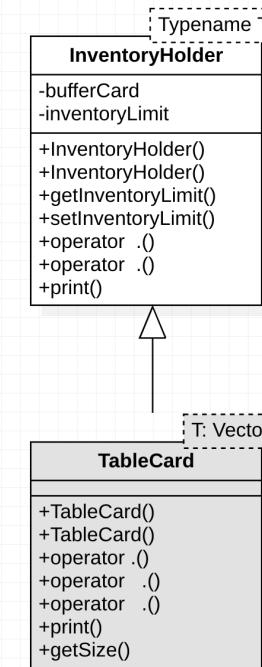
## Diagram Inheritance Kelas InventoryHolder



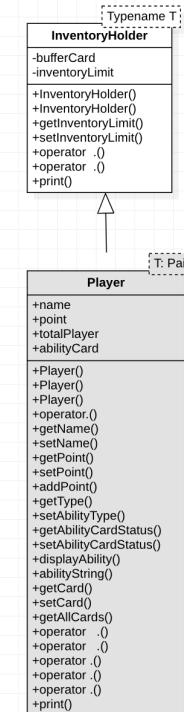
# Diagram Inheritance Kelas Deck



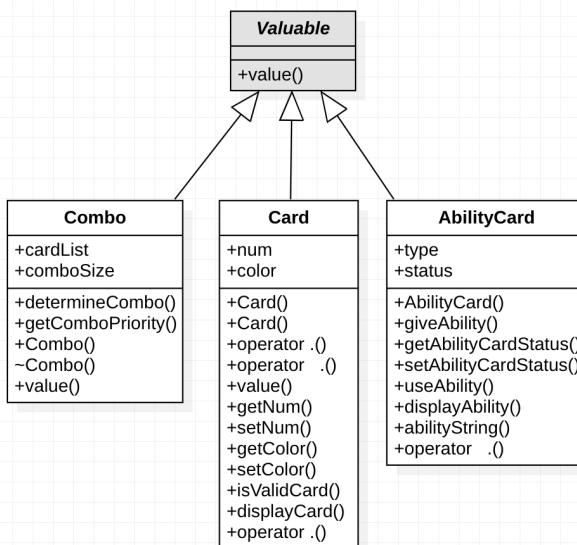
## Diagram Inheritance Kelas TableCard



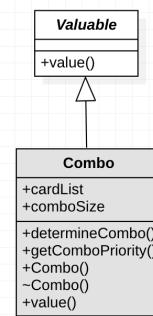
## Diagram Inheritance Kelas Player



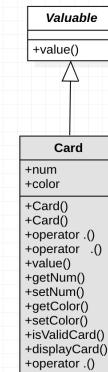
### Inheritance Diagram Kelas Valuable



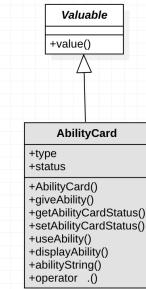
### Inheritance Diagram Kelas Combo



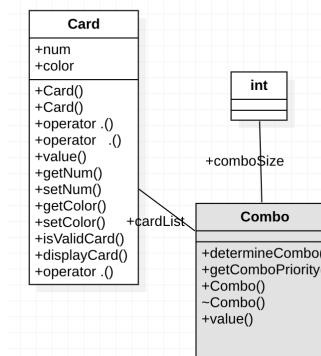
### Inheritance Diagram Kelas Card



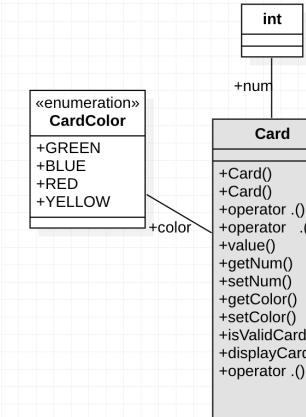
### Inheritance Diagram Kelas AbilityCard



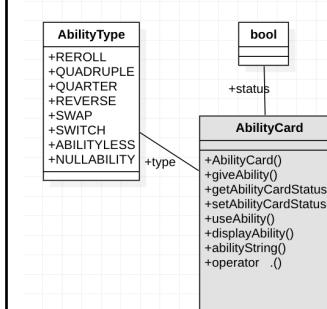
### Collaboration Diagram Kelas Combo



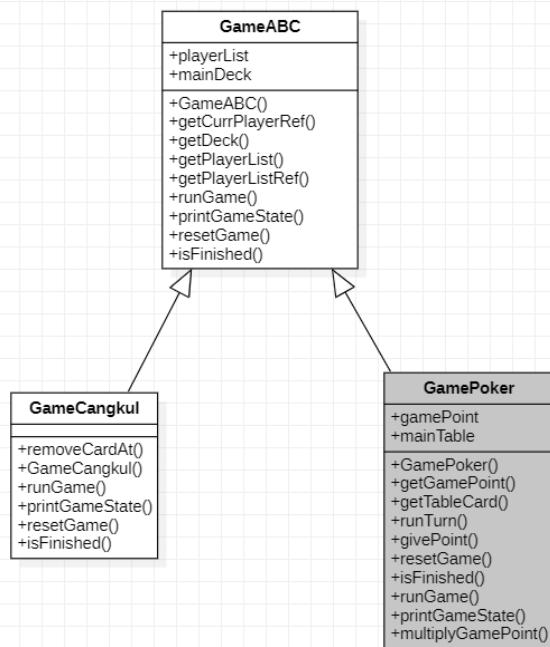
### Collaboration Diagram Kelas Card



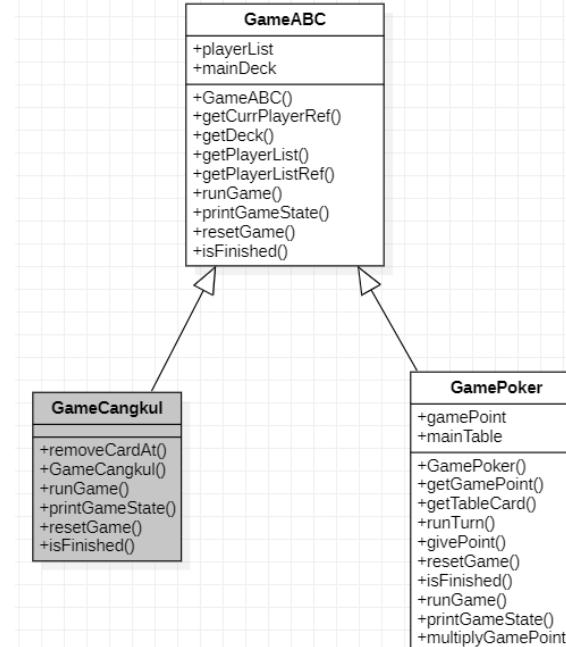
### Collaboration Diagram Kelas AbilityCard



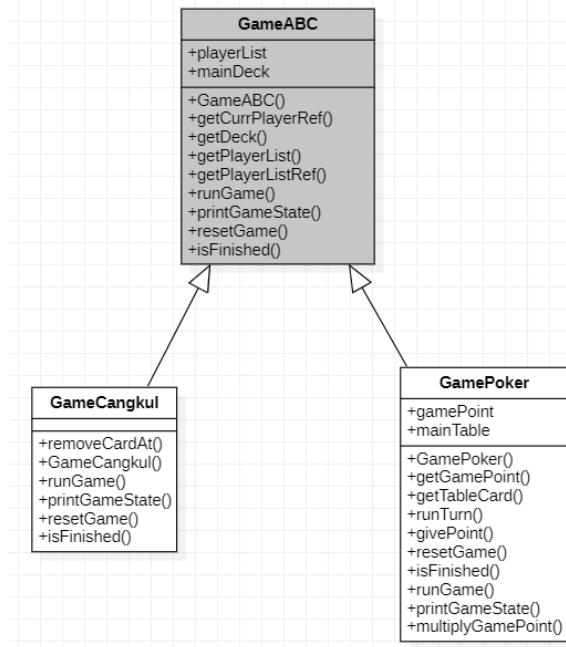
### Inheritance Diagram Kelas GamePoker

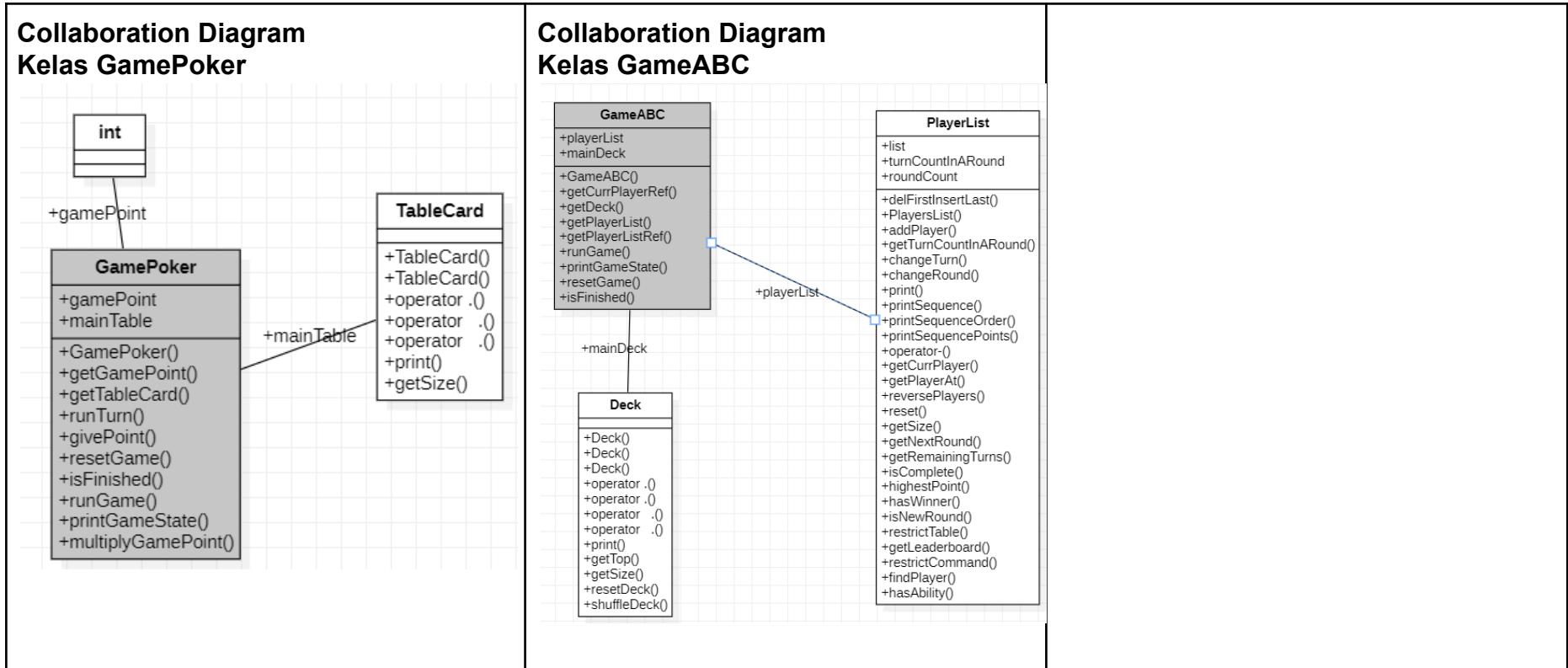


### Inheritance Diagram Kelas GameCangkul



### Inheritance Diagram Kelas GameABC





### 1.3. Desain Kelas

#### A. Command

Kelas Command merupakan kelas yang mengatasi *command* atau perintah dari Player. Kelas ini memanfaatkan prinsip polimorfisme, sehingga pada Implementasinya hanya memanggil kelas basis command. Pada kelas command, terdapat 2 kelas anak: AbilityCommand dan BasicCommand. Keduanya mempunyai anak kelas yang mengatur *command*/perintah secara spesifik (seperti Double, Reroll, Next). Kelas Command dibuat sehingga penambahan command menjadi lebih mudah dan flexibel karena cukup ditambahkan pada kelas tersebut.

#### B. Exception

Kelas Exception merupakan kelas yang menangani berbagai jenis pengecualian, seperti validasi pada *array*, *command*, *config*, *input*, dan *game*. Kelas ini menggunakan konsep polimorfisme dan *inheritance* dalam implementasinya. Hal ini dikarenakan kelas ini hanya perlu menggeneralisasi tipe objek pada kelas anak yang diinstansiasi. Pada kelas ini, dapat dilakukan catch Exception berdasarkan tipenya. Karena menggunakan prinsip polimorfisme, cukup ditangkap *exception* pada kelas *parentnya* tetapi memungkinkan exception yang lebih spesifik untuk ditangkap, seperti *InputOutOfRangeException*. Implementasi ini juga lebih mudah dimanfaatkan dalam sebuah *try catch block*.

#### C. Game

Kelas Game merupakan kelas yang mengimplementasikan mekanisme permainan. Kelas ini menggunakan konsep composability, dimana di dalamnya terdapat kelas PlayersList, Deck, dan TableCard. Pada kelas ini, terdapat method yang krusial bagi keberjalanannya game, seperti *runGame()*, *resetGame()*, dan *runTurn()*.

#### D. InventoryHolder

InventoryHolder menggunakan konsep *inheritance*, dengan Player, TableCard, dan Deck sebagai kelas anaknya. InventoryHolder merupakan kelas untuk menyimpan objek benda yang diolah pada permainan. Kelas ini menerapkan konsep generic class, digunakan untuk menggeneralisasi tipe objek pada kelas anak yang diinstansiasi (termasuk tipe *vector* dan *pair*). Kelas ini juga menerapkan konsep operator overloading (misalnya *>>* berarti menambah kartu ke dalam InventoryHolder dan *<<* berarti mengurangi kartu dari InventoryHolder).

#### E. Playerslist

PlayersList menggunakan STL Collection vector. Kelas ini digunakan untuk mengatur player-player dan penggantian putaran Player pada permainan. Pada kelas ini, digunakan juga konsep operator overloading. Operator '-' digunakan untuk mengurangi seorang Player dari PlayerList.

#### F. Randomizer

Randomizer merupakan kelas yang digunakan untuk mengacak suatu array. Kelas ini memanfaatkan konsep generic/template function sehingga array dapat bertipe macam-macam.

#### G. Valuables

Kelas Valuables memanfaatkan konsep *abstract inheritance*, dengan kelas Combo dan Card sebagai instansiasi anaknya. Kelas ini bertujuan untuk mendapatkan nilai atau *value* dari Card (satuan), dan Combo (jika Card tersebut digabungkan). Penggunaan kelas ini dapat memudahkan fleksibilitas keberjalanannya program karena jika ingin membuat mekanisme lainnya (seperti Capsa) dapat mengubah hanya kelas

ini. Kelas Card mendeskripsikan objek kartu yang dipakai, termasuk nomor dan tipe kartu. Kelas Combo mendeskripsikan kombo yang dihasilkan dari berbagai variasi kartu.

#### H. Input Handler

Kelas input handler merupakan kelas yang menangani input dan masukan untuk program. Kelas ini bertujuan untuk mendapatkan exception sehingga program tidak akan keluar dengan abnormal.

#### I. Algorithm

Kelas algorithm merupakan kelas yang menangani algoritma yang terdapat pada program, termasuk algoritma randomizer dan maxValue. Kelas ini menggunakan konsep template sehingga randomizer maupun maxValue dapat digunakan dengan tipe yang berbeda. Pada program ini, digunakan tipe Vector, Card, Combo, dan Player.

#### J. Combine

Kelas combine merupakan kelas yang menangani kemungkinan kombinasi dari seluruh kartu pemain dan kartu di meja (TableCard).

## 2. Penerapan Konsep OOP

### 2.1. Inheritance & Polymorphism

Konsep *inheritance* digunakan dalam beberapa kelas pada program. Kelas yang menggunakan *inheritance* meliputi: Exception dengan kelas anaknya yang meliputi semua exception, InventoryHolder (Parent) dengan Deck (child), Player (child) dan TableCard (child), Valuables (Parent) dengan Card (Child) dan Combo (Child), Command dengan kelas anaknya yang meliputi semua command. Dengan penggunaan *inheritance*, dimungkinkan pemanfaatan konsep polimorfisme. Polimorfisme memiliki peran penting dalam tipe dasar kelasnya. Konsep ini sangat penting dalam penggunaan kelas Command dan Exception. Pada kelas Command, terdapat satu fungsi utama dengan signature yang sama bernama execute. Walaupun begitu, dengan konsep polimorfisme, perintah yang dieksekusi berbeda-beda berdasarkan command yang dimasukkan Player. Pada kelas Exception, terdapat satu fungsi utama bernama what yang dapat mengembalikan *error message* berdasarkan *error* yang dithrow.

```

#ifndef COMMAND_HPP
#define COMMAND_HPP

#include <iostream>
#include <string>
#include "../game/game.hpp"
using namespace std;

class Command {
protected:
    Game* game;
public:
    Command(Game*);
    virtual void execute() = 0;
    virtual ~Command();
};

#endif

#ifndef BASIC_COMMAND_H
#define BASIC_COMMAND_H

#include "command.hpp"

class BasicCommand : public Command {
public:
    BasicCommand(Game*);
    virtual void execute() = 0;
    virtual ~BasicCommand();
};

class Double : public BasicCommand{
public:
    Double(Game*);
    void execute();
    ~Double();
};

class Half : public BasicCommand{
public:
    Half(Game*);
    void execute();
    ~Half();
};

class Next : public BasicCommand{
public:
    Next(Game*);
    void execute();
    ~Next();
};

#endif

#ifndef ABILITY_COMMAND_HPP
#define ABILITY_COMMAND_HPP

#include <iostream>
#include "command.hpp"
using namespace std;

class AbilityCommand : public Command {
public:
    AbilityCommand(Game*);
    virtual void execute() = 0;
    void turnOffAbility();
    virtual ~AbilityCommand();
};

class Reroll : public AbilityCommand {
public:
    Reroll(Game*);
    void execute();
    ~Reroll();
};

class Quadruple : public AbilityCommand {
public:
    Quadruple(Game*);
    void execute();
    ~Quadruple();
};

class Quarter : public AbilityCommand {
public:
    Quarter(Game*);
    void execute();
    ~Quarter();
};

class Abilityless : public AbilityCommand {
public:
    Abilityless(Game*);
    void execute();
    ~Abilityless();
};

#endif

class ReverseDirection : public AbilityCommand {
public:
    ReverseDirection(Game*);
    void execute();
    ~ReverseDirection();
};

class SwapCard : public AbilityCommand {
public:
    SwapCard(Game*);
    void execute();
    ~SwapCard();
};

class SwitchCard : public AbilityCommand {
public:
    SwitchCard(Game*);
    void execute();
    ~SwitchCard();
};

```

Dari kiri ke kanan: Kelas Command (Parent), Kelas Basic Command (Child), Kelas Ability Command (Child), lanjutan dari Kelas Ability

## 2.2. Method/Operator Overloading

Konsep operator overloading digunakan di kelas Inventory Holder, Card, TableDeck, dan Player. Metode ini diimplementasikan ketika sebuah fungsi mempunyai logika penggunaan yang sama dengan sebuah operator. Konsep ini juga digunakan untuk meningkatkan *readability code*. Sebagai contoh, alih-alih menggunakan fungsi doesPlayerHaveLessPoints() pada Player, dapat digunakan operator <. Hal yang sama berlaku ketika ingin mengurangi kartu pada *deck*, dapat memakai operator '<<'. Pada kelas ini juga digunakan metode *function override*, yang berguna untuk mengambil alih suatu fungsi dari fungsi yang dimiliki *parent*.

```
#ifndef INVENTORY HOLDER_HPP
#define INVENTORY HOLDER_HPP

#include <iostream>
#include <type_traits>
#include <vector>

#include "../valuables/card.hpp"
using namespace std;

...
template <typename T>
class InventoryHolder {
protected:
    T bufferCard;
    int inventoryLimit;

public:
    // ctor-cctor-dtor
    InventoryHolder();
    InventoryHolder(int inventoryLimit);

    int getInventoryLimit();
    void setInventoryLimit(int inventoryLimit);

    // operators
    virtual InventoryHolder& operator<<(const Card& card) = 0;
    virtual InventoryHolder& operator>>(Card* card) = 0;
    virtual void print() = 0;
};
```

```
class Player : public InventoryHolder<pair<Card, Card> > {
private:
    string name;
    long int point;
    static int totalPlayer;
    AbilityType abilityCard;

public:
    // ctor-cctor-dtor
    Player();
    Player(string name, long int point);
    Player(const Player& other);
    Player& operator=(const Player& other);

    // services
    string getName() const;
    void setName(string name);

    long int getPoint() const;
    void setPoint(long int point);
    void addPoint(long int point);

    // Ability Card
    AbilityType getType() const;
    void setAbilityType(AbilityType type);
    bool getAbilityCardStatus() const;
    void setAbilityCardStatus(bool status);
    void displayAbility();

    // IDX 0 for LeftCard, IDX 1 for RightCard
    Card getCard(int idx);
    void setCard(int idx, Card card);
};
```

```
Card* getAllCards() const;
// resetCards();
// void setCards(Card* card);

// operator
Player& operator<<(const Card& card) override;
Player& operator>>(Card* card) override;

bool operator<(const Player& other) const;
bool operator>(const Player& other) const;
bool operator==(const Player& other) const;

void print() override;
};

#endif
```

Kotak ungu merepresentasikan operator,kotak hijau merepresentasikan override.

Dari kiri ke kanan: Kelas InventoryHolder, Kelas Player, lanjutan dari kelas Player.

## 2.3. Template & Generic Classes

Konsep template dan Generic Classes digunakan pada kelas InventoryHolder, Deck, Player, dan TableCard. InventoryHolder yang merupakan kelas *parent* dari ketiganya memiliki atribut bufferCard yang bertipe T. Dengan adanya template ini, tidak perlu didefinisikan kembali ketika tipe elemen yang dibutuhkan oleh kelas berbeda. Abstraksi terhadap atribut dan metode juga dapat dilakukan menggunakan tipe yang disediakan oleh template. Pada InventoryHolder, digunakan 2 tipe data yang berbeda yaitu *vector* dan *pair*.

```
#ifndef INVENTORY HOLDER_HPP
#define INVENTORY HOLDER_HPP

#include <iostream>
#include <type_traits>
#include <vector>

#include "../valuables/card.hpp"
using namespace std;

template <typename T>
class InventoryHolder {
protected:
    T bufferCard;
    int inventoryLimit;

public:
    // ctor=ctor-dtor
    InventoryHolder();
    InventoryHolder(int inventoryLimit);

    int getInventoryLimit();
    void setInventoryLimit(int inventoryLimit);

    // operators
    virtual InventoryHolder& operator<<(const Card& card) = 0;
    virtual InventoryHolder& operator>>(Card* card) = 0;
    virtual void print() = 0;
};

#endif
```

```
#ifndef DECK_HPP
#define DECK_HPP

#include <algorithm>
#include <iostream>
#include <vector>

#include "../valuables/card.hpp"
#include "inventory_holder.hpp"
using namespace std;

class Deck : public InventoryHolder<vector<Card> > {
public:
    // General Methods
    Deck(); // ctor
    Deck(const Card& card);
    Deck(const Deck& other);
    Deck& operator=(const vector<Card> & card);
    Deck& operator=(const Deck& other);

    Deck& operator<<(const Card& card) override;
    Deck& operator>>(Card* card) override;
    void print() override;

    // Methods
    Card getTop() const;
    int getSize() const;
    // Card* getTwo() const;
    void resetDeck();
    void shuffleDeck();
};

#endif
```

```
class Player : public InventoryHolder<pair<Card, Card> > {
private:
    string name;
    long int point;
    static int totalPlayer;
    AbilityType abilityCard;

public:
    // ctor=ctor-dtor
    Player();
    Player(string name, long int point);
    Player(const Player& other);
    Player& operator=(const Player& other);

    // services
    string getName() const;
    void setName(string name);

    long int getPoint() const;
    void setPoint(long int point);
    void addPoint(long int point);

    // Ability Card
    AbilityType getType() const;
    void setAbilityType(AbilityType type);
    bool getAbilityCardStatus() const;
    void setAbilityCardStatus(bool status);
    void displayAbility();

    // IDX 0 for LeftCard, IDX 1 for RightCard
    Card getCard(int idx);
    void setCard(int idx, Card card);
};

#endif
```

```
#ifndef TABLE_CARD_HPP
#define TABLE_CARD_HPP

#include <iostream>
#include <string>

#include "../valuables/card.hpp"
#include "inventory_holder.hpp"

class TableCard : public InventoryHolder <vector <Card> > {
public:
    // ctor=ctor-dtor
    TableCard();
    TableCard(const TableCard& other);
    TableCard& operator=(const TableCard& other);

    TableCard& operator<<(const Card& card) override;
    TableCard& operator>>(Card* card) override;

    // services
    Card* getAllCards();
    void print() override;

    int getSize() const;
};

#endif
```

Kotak jingga menekankan template dan instansiasi dari kelas template.

Dari kiri ke kanan: Kelas InventoryHolder, Kelas Deck, kelas Player, dan kelas TableCard.

## 2.4. Exception

Konsep Exception digunakan hampir di setiap aspek program, terutama jika program meminta input pada pengguna. Kelas ini memungkinkan adanya penanganan kesalahan yang muncul saat eksekusi program. Ini menghindari terjadinya pemberhentian program secara abnormal. Dengan menggunakan blok try, catch, dan throw, exception dapat ditangkap dan diulang prosesnya (direvisi) jika perlu. Pada program ini,

terdapat 5 jenis exception yaitu: exception pada array, masukan command, config, pada mekanisme game, dan pada masukan user. Kelima jenis tersebut merupakan kelas tersendiri yang mempunyai *parent class* Exception sehingga blok try, catch, throw bisa digeneralisasi saat di-catch.

```

...
#ifndef EXCEPTION_H
#define EXCEPTION_H

#include <string>

class Exception{
protected:
    const std::string message;
public:
    virtual const std::string what() const throw() = 0;
};

#endif

#ifndef CONFIG_EXCEPTION_H
#define CONFIG_EXCEPTION_H

#include "exception.hpp"

class ConfigException : public Exception{
protected:
    const std::string message = "Invalid Config operation";
public:
    virtual const std::string what() const throw() {return message;};
};

//ini buat input nomor yang gak valid (angka diisi huruf, kalo redundan hapus aja
class ConfigFileNotFoundException : public ConfigException{
private:
    const std::string message = "Error opening file";
public:
    const std::string what() const throw() {return message;};
};

//ini buat input nomor yang gak valid (angka diisi huruf, kalo redundan hapus aja
class ConfigInvalidCharException : public ConfigException{
private:
    const std::string message = "Unexpected char in config";
public:
    const std::string what() const throw() {return message;};
};

//ini buat input warna yang gak valid
class ConfigInvalidColorException : public ConfigException{
private:
    const std::string message = "Card color in config is invalid";
public:
    const std::string what() const throw() {return message;};
};

#endif

#ifndef ARRAY_EXCEPTION_H
#define ARRAY_EXCEPTION_H

#include "exception.hpp"

class ArrayException : public Exception{
protected:
    const std::string message = "Invalid array operation";
public:
    virtual const std::string what() const throw() {return message;};
};

class ArrayFull : public ArrayException{
private:
    const std::string message = "Array is full";
public:
    const std::string what() const throw() {return message;};
};

class ArrayEmpty : public ArrayException{
private:
    const std::string message = "Array is Empty";
public:
    const std::string what() const throw() {return message;};
};

class ArrayIndexInvalid : public ArrayException{
private:
    const std::string message = "Array index is invalid";
public:
    const std::string what() const throw() {return message;};
};

#endif

#ifndef COMMAND_EXCEPTION_H
#define COMMAND_EXCEPTION_H

#include "exception.hpp"

class CommandException : public Exception{
protected:
    const std::string message = "Invalid command";
public:
    virtual const std::string what() const throw() {return message;};
};

class CommandNotAvailable : public CommandException{
private:
    const std::string message = "Ability not available";
public:
    const std::string what() const throw() {return message;};
};

class CommandInvalid : public CommandException{
private:
    const std::string message = "Ability is invalid";
public:
    const std::string what() const throw() {return message;};
};

#endif

#ifndef GAME_EXCEPTION_H
#define GAME_EXCEPTION_H

#include "exception.hpp"

class GameException : public Exception{
protected:
    const std::string message = "Invalid game operation";
public:
    virtual const std::string what() const throw() {return message;};
};

class GameMultiplierException : public GameException{
private:
    const std::string message = "Invalid point multiplier";
public:
    const std::string what() const throw() {return message;};
};

#endif

#ifndef INPUT_EXCEPTION_H
#define INPUT_EXCEPTION_H

#include "exception.hpp"

class InputException : public Exception{
private:
    const std::string message = "Invalid input";
public:
    virtual const std::string what() const throw() {return message;};
};

class InputInvalidCharException : public InputException{
private:
    const std::string message = "Unexpected char in input";
public:
    const std::string what() const throw() {return message;};
};

class InputOutOfRangeException : public InputException{
private:
    const std::string message = "Input is out of range";
public:
    const std::string what() const throw() {return message;};
};

class InputUnavailableException : public InputException{
private:
    const std::string message = "Input is not available";
public:
    const std::string what() const throw() {return message;};
};

#endif

```

## 2.5. C++ Standard Template Library

C++ STL banyak digunakan pada setiap kelas. STL digunakan untuk memudahkan implementasi fungsi sehingga tidak perlu mengimplementasikan fungsi sendiri. Berikut merupakan keterangan dari setiap fungsi yang dipakai pada program.

|                |  |
|----------------|--|
| 1. <fstream>   | Library ini digunakan untuk membaca file dan memasukkan data ke dalam suatu tipe.  |
| 2. <algorithm> | Library ini digunakan untuk men- <i>sorting</i> pada vektor.                       |
| 3. <Vector>    | Library ini digunakan untuk tipe data pada kelas Deck, TableCard, dan PlayersList. |
| 4. <Pair>      | Library ini digunakan untuk tipe bufferCard pada kelas Player.                     |
| 5. <String>    | Library ini banyak digunakan hampir pada setiap kelas untuk tipe data string.      |
| 6. <random>    | Library ini berfungsi untuk mengacak sebuah array/list.                            |
| 7. <iostream>  | Library ini berfungsi sebagai fungsi input/output.                                 |
| 8. <math>      | Library ini berfungsi untuk kalkulasi matematis.                                   |
| 9. <map>       | Library ini digunakan untuk <i>mapping</i> enumerasi pada Ability dan Card.        |

```
#ifndef PLAYERSLIST_HPP
#define PLAYERSLIST_HPP

#include "../inventory_holder/player.hpp"
#include <iostream>
#include <algorithm>
#include <math.h>
#include <vector>

using namespace std;

You, 19 minutes ago | 4 authors (You and others)
class PlayersList {
private:
    vector<Player> list;
    int turnCountInARound, roundCount;

    void delFirstInsertLast();

public:
    PlayersList();
    void addPlayer(Player p);
    int getRoundCount() const;
    int getTurnCountInARound() const;
    void changeTurn();
    void changeRound();
    void print();
    void printSequence();
    void printSequenceOrder();
    void printSequencePoints();

    PlayersList operator-(const Player& other);

    Player& getCurrPlayer();
    Player& getPlayerAt(int i);
```

Contoh kelas PlayersList yang memakai <algorithm>, <iostream>, <math>, dan <vector>

## 2.6. Konsep OOP lain

Berikut merupakan konsep OOP lainnya yang digunakan pada program ini.

|  |                            |   |
|--|----------------------------|---|
| <pre>#ifndef VALUABLE_HPP #define VALUABLE_HPP  ... class Valuable { public:     // Get the value of this valuable     virtual float value() = 0; };  #endif </pre>  | <p>Abstract Base Class</p> | <p>Konsep Abstract Base Class digunakan pada kelas Command, Exception, dan Valuable. Pada kelas-kelas ini, <i>parent class</i> tidak bisa diinstansiasi. Misalnya kelas valuable hanya mengandung satu <i>virtual method</i> value().</p> |
| <pre>#ifndef COMMAND_HPP #define COMMAND_HPP  #include &lt;iostream&gt; #include &lt;string&gt; #include "../game/game.hpp" using namespace std;  class Command { protected:     Game* game; public:     Command(Game*);     virtual void execute() = 0;     virtual ~Command(); };  #endif </pre> | <p>Composition</p>         | <p>Konsep Composition digunakan pada kelas Command, Combo, PlayersList. Pada kelas ini, kelas Game merupakan atribut dari Command. Kelas Players merupakan atribut dari PlayersList.</p>  |
| <pre>#ifndef TABLE_CARD_HPP #define TABLE_CARD_HPP  #include &lt;iostream&gt; #include &lt;string&gt;  #include "../valuables/card.hpp" #include "./inventory_holder.hpp"  ... class TableCard : public InventoryHolder &lt;vector &lt;Card&gt; &gt; { public:     ... }</pre>                     | <p>Collection</p>          | <p>Konsep collection banyak digunakan pada program ini. Contohnya adalah pemakaian pair dan vector pada kelas Player dan Deck.</p>  |

### 3. Bonus Yang dikerjakan

#### 3.1. Bonus yang diusulkan oleh spek

##### 3.1.1. Template/Generic Function

Template atau Generic Function digunakan pada kelas Randomizer. Kelas randomizer memungkinkan untuk mengacak urutan dari sebuah array. Randomizer diimplementasikan untuk tipe data int, dan Card. Template Class sudah dijelaskan pada subbab sebelumnya.

```
#ifndef RANDOMIZER_H
#define RANDOMIZER_H
#include <random>

...
class Randomizer{
public:
    template<typename T>
    void iterShuffle(T array[], int n);

    template<typename T>
    void iterShuffle(T* array);

    template<typename T>
    T removeRandom(T* array);

    template<typename T>
    T pickRandom(T* array);
};

#endif
```

```
template<typename T>
void Randomizer::iterShuffle(T array[], int n) {
    std::random_device seed;
    std::mt19937 g(seed());
    std::shuffle(&array[0], &array[n], g);
}

template<typename T>
void Randomizer::iterShuffle(T* array) {
    std::random_device seed;
    std::mt19937 g(seed());
    std::shuffle(array->begin(), array->end(), g);
}

template<typename T>
T Randomizer::removeRandom(T array[]){
    iterShuffle(array);
    T temp = array->begin();
    *array->erase(array->begin());
    return temp;
}

template<typename T>
T Randomizer::pickRandom(T array[]){
    iterShuffle(array);
    T temp = array->begin();
    return temp;
}
```

## 3.2. Bonus Kreasi Mandiri

### 3.2.1. Unit Testing Implementation

Dalam rangka meningkatkan efisiensi dan efektivitas kerja, terdapat implementasi unit testing yang berfungsi untuk mengetes dan mengecek setiap modul yang ada pada program. Untuk setiap modul yang berisi fungsi eksekusi terdapat modul yang mengecek implementasi fungsi tersebut.

```

int main() {
    // DECK TESTING
    Deck deck;
    deck.print();
    cout << "SHUFFLE\n";
    deck.shuffleDeck();
    deck.print();

    cout << "TOOK CARD 1 BY 1\n";
    int currSize = deck.getSize();
    for (int i = 0; i < currSize; i++) {
        Card temp;
        deck >> &temp;
        temp.displayCard();
    }

    deck.print();

    cout << "=====\\n";

    cout << "INSERT/DELETE DECK\\n";
    deck.resetDeck();
    cout << "SIZE: " << deck.getSize() << endl;
    cout << "----- RESET DECK -----\\n";
    deck.print();
    Card card1;
    Card card2;
    deck >> &card1;
    deck >> &card2;
    deck.getTop().displayCard();
    card1.displayCard();
    card2.displayCard();
    deck << Card(1, CardColor::BLUE); // try catch dulu, kartu harus unik
    deck.getTop().displayCard();
    ...
}

int main() {
    PlayersList p;
    string name;
    // Check input
    for (int i = 0; i < 7; i++) {
        cin >> name;
        p.addPlayer(Player(name, pow(2, 30 + i)));
    }

    cout << "===== BASED p INPUT =====" << endl;
    p.print();
    cout << endl << endl;

    // copy
    PlayersList copy = p;

    // change turn
    // tc = 0
    p.changeTurn();
    // tc = 1
    p.changeTurn();
    p.changeTurn();
    p.changeTurn();
    p.changeTurn();
    p.changeTurn();
    p.changeTurn();

    // tc = 2
    cout << "===== CHANGE p TURN =====" << endl;
    p.print();
    cout << endl << endl;

    cout << "===== UNDO CHANGE p TURN =====" << endl;
    p.undoChangeTurn();
    p.print();
    cout << endl << endl;

    // remaining turns
    cout << "===== p REMAINING TURNS =====" << endl;
    p.getRemainingTurns().print();
    cout << endl << endl;
}

int main() {
    // Shuffle deck
    srand(time(nullptr));
    vector<int> deck;
    for (int i = 1; i <= 52; i++) {
        deck.push_back(i);
    }
    random_shuffle(deck.begin(), deck.end());

    // Create random combo
    Card** cardList = new Card*[10];
    int j = 0;
    for (int i = 0; i < 10; i++) {
        cardList[i] = new Card[5];
        for (int k = 0; k < 5; k++) {
            for (int l = 0; l < 52; l++) {
                int num = (deck[j] - 1) % 13 + 1;
                CardColor color = static_cast<CardColor>((deck[j] - 1) / 13);
                cardList[i][k] = *(new Card(num, color));
                j++;
            }
        }
    }

    // Calculate combo value
    float comboValues[10];
    int comboPrio[10];
    for (int i = 0; i < 10; i++) {
        Combo temp(cardList[i], 5);
        comboValues[i] = temp.value();
    }
}

```

### 3.2.2. Tampilan Program

Dalam rangka meningkatkan keterbacaan program bagi pemain, terdapat implementasi tampilan antarmuka yang berfungsi untuk menampilkan kartu serta informasi relevan yang terdapat pada program.

```
----- TABLE CARD -----
10 Biru      4 Hijau     12 Hijau    6 Kuning    6 Merah
[10]          [4]          [12]         [6]          [6]
               *             *             *             *
               10            4             12            6             6

Sekarang giliran 1!
Game Point : 64

-----
Point   : 0
Ability  : Kartu REVERSE (belum digunakan)
Cards   : 1 Merah, 4 Kuning

-----
----- PLAYER CARD -----
1 Merah      4 Kuning
[1]          [4]
               *             *
               1             4

-----
Masukkan command
> 
```

### 3.2.3. Fitur Help

Dalam rangka menangani banyak ambiguitas pada spek, kami menambahkan fitur help untuk membantu pengguna dan pemain dalam menerapkan gameplay.

```
void Help::execute() {
    // Interface
    cout << "\033[1m\033[36m \n";
    cout << "-----*** COMMANDS GUIDE ***-----" << endl;
    cout << "|";
    cout << "-----* BASIC COMMANDS *-----" << endl;
    cout << "          Dapat dipanggil kapan saja" << endl;
    cout << "NEXT      : Tidak melakukan apa-apa" << endl;
    cout << "DOUBLE    : Mengalikan poin pada meja menjadi dua kali lipat" << endl;
    cout << "|";
    cout << "-----* ABILITY COMMANDS *-----" << endl;
    cout << "          Dapat dipanggil jika memiliki kartu ability yang bersesuaian" << endl;
    cout << "HALF      : Mengalikan poin pada meja menjadi setengah kali lipat" << endl;
    cout << "REROLL    : Mengambil ulang 2 kartu" << endl;
    cout << "QUARTER   : Mengalikan poin pada meja menjadi seperempat kali lipat" << endl;
    cout << "QUADRUPLE : Mengalikan poin pada meja menjadi empat kali lipat" << endl;
    cout << "ABILITYLESS : Mematikan kartu ability milik player lain" << endl;
    cout << "REVERSE   : Memutar arah giliran" << endl;
    cout << "SWAPCARD  : Menukar 1 kartu milik pemain lain dengan 1 kartu pemain lainnya" << endl;
    cout << "SWITCH    : Menukar 2 kartu milik sendiri dengan pemain lain" << endl;
    cout << "HELP      : Memberikan list command yang dapat diberikan" << endl;
    cout << "|";
    cout << "-----" << endl;
    cout << "\033[0m";
}
```

#### 4. Pembagian Tugas

| Modul (dalam poin spek) |                   | Implementer                  | Tester                       |
|-------------------------|-------------------|------------------------------|------------------------------|
| Command                 | ability_command   | 13521129, 13521171, 13521151 | 13521129                     |
|                         | basic_command     | 13521129                     | 13521129                     |
|                         | command           | 13521129                     | 13521129                     |
| Exception               | array_exception   | 13521095                     | 13521095                     |
|                         | command_Exception | 13521095                     | 13521095                     |
|                         | config_exception  | 13521095                     | 13521095                     |
|                         | game_exception    | 13521095                     | 13521095                     |
|                         | input_exception   | 13521095                     | 13521095                     |
| game                    |                   | 13521129                     | 13521129                     |
| input_handler           | input_handler     | 13521149                     | 13521149                     |
|                         | file_handler      | 13521151                     | 13521151, 13521095, 13521171 |
|                         | command_handler   | 13521095                     | 13521095                     |
| inventory_holder        | deck              | 13521171                     | 13521149<br>13521171         |
|                         | player            | 13521171                     | 13521149<br>13521171         |

|             |                  |          |                      |
|-------------|------------------|----------|----------------------|
|             | table_card       | 13521171 | 13521149<br>13521171 |
|             | inventory_holder | 13521171 | 13521149<br>13521171 |
| playerslist |                  | 13521129 | 13521129             |
| randomizer  |                  | 13521095 | 13521095             |
| valuables   | card             | 13521151 | 13521151             |
|             | combo            | 13521149 | 13521149             |
|             | valuable         | 13521149 | 13521149             |
| Combine     |                  | 13521129 | 13521129             |
| Runner      |                  | 13521095 | 13521095             |

## Lampiran Foto Kelompok

