

**1. discardPile**

* **Untuk Apa:**
  + **discardPile** atau tumpukan kartu buangan adalah tempat di mana kartu yang telah dimainkan oleh pemain diletakkan. Kartu teratas di tumpukan ini menentukan aturan permainan saat ini, seperti warna atau angka yang harus diikuti oleh pemain berikutnya.
* **Fungsi:**
  + Melacak kartu yang telah dimainkan.
  + Menentukan aturan permainan saat ini berdasarkan kartu teratas.
  + Menyediakan dasar untuk validasi langkah pemain (apakah kartu yang dimainkan valid).

**2. currentPlayer**

* **Untuk Apa:**
  + **currentPlayer** menunjuk ke pemain yang saat ini sedang giliran bermain. Ini membantu mengatur alur permainan, memastikan bahwa hanya pemain yang tepat yang bisa melakukan tindakan seperti menarik kartu atau memainkan kartu.
* **Fungsi:**
  + Mengidentifikasi pemain yang saat ini sedang dalam giliran.
  + Mengontrol giliran dan mengelola urutan permainan.
  + Memastikan bahwa aksi permainan dilakukan oleh pemain yang benar.

**3. nextTurn**

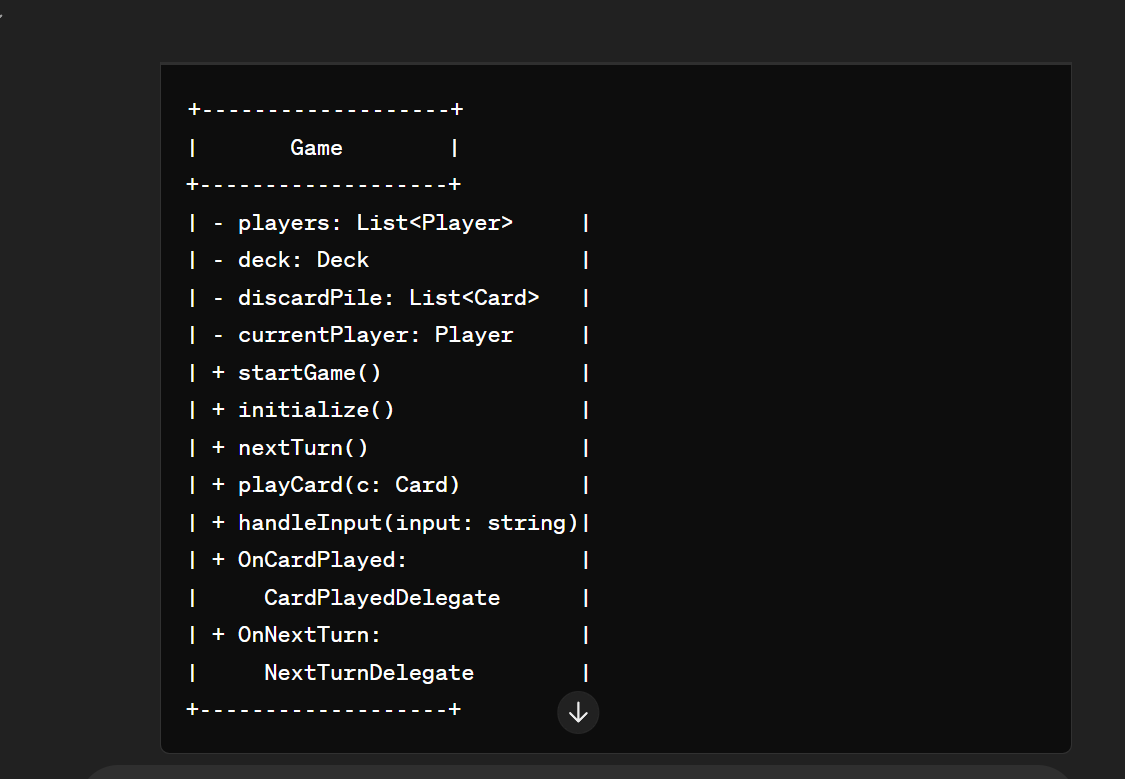
* **Untuk Apa:**
  + **nextTurn** adalah metode yang digunakan untuk mengatur giliran ke pemain berikutnya setelah pemain saat ini menyelesaikan gilirannya.
* **Fungsi:**
  + Mengatur dan memindahkan giliran ke pemain berikutnya dalam urutan.
  + Menjaga alur permainan agar tetap berjalan lancar dan adil.
  + Menangani perubahan giliran berdasarkan aksi kartu khusus seperti **Reverse**.

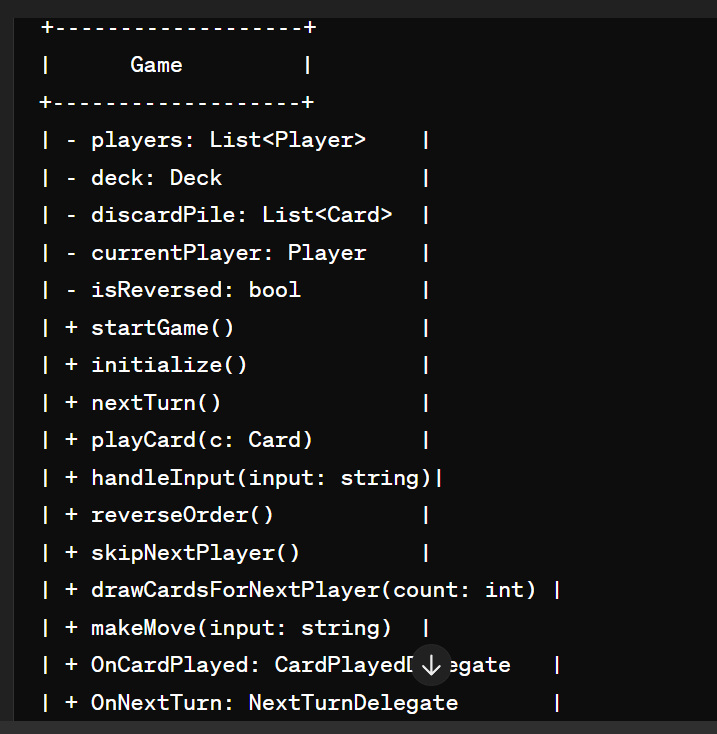
**4. playCard**

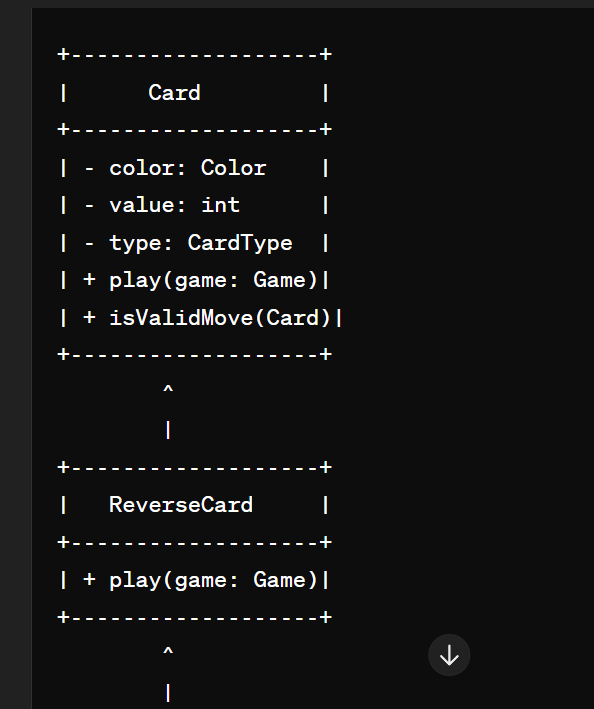
* **Untuk Apa:**
  + **playCard** adalah metode yang digunakan untuk memainkan kartu dari tangan pemain ke tumpukan kartu buangan. Ini melibatkan validasi apakah kartu tersebut bisa dimainkan berdasarkan aturan permainan saat ini.
* **Fungsi:**
  + Memindahkan kartu dari tangan pemain ke tumpukan buangan.
  + Memvalidasi apakah kartu yang dimainkan sesuai dengan kartu teratas di **discardPile**.
  + Menjalankan efek kartu khusus (misalnya, **Skip**, **Draw Two**, **Reverse**).
  + Memeriksa apakah pemain telah memenangkan permainan setelah memainkan kartu.

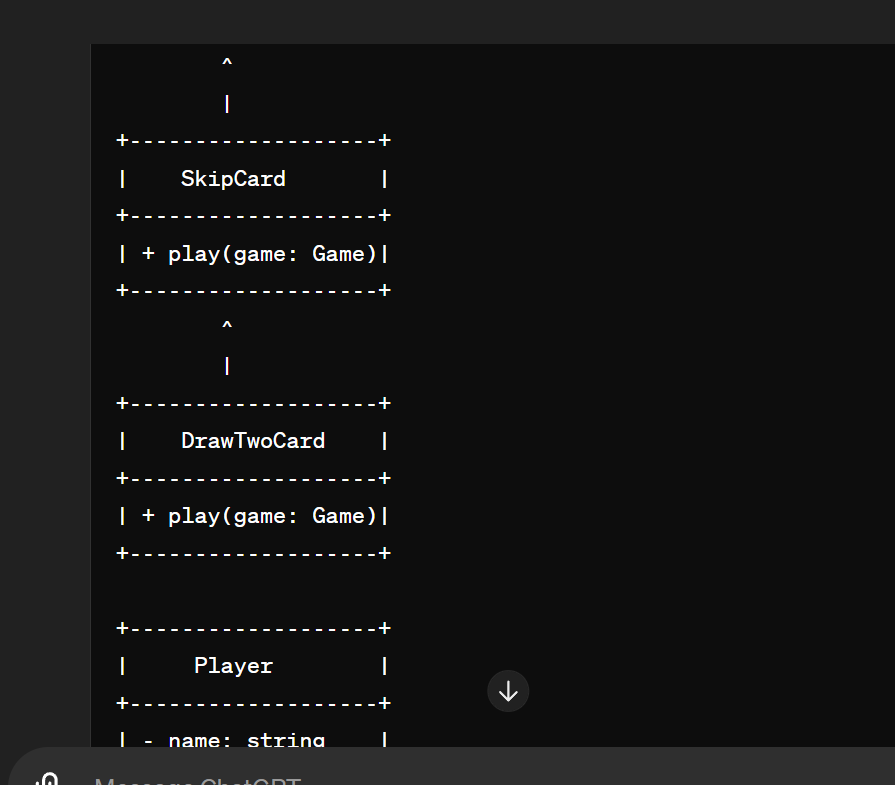
**5. handleInput**

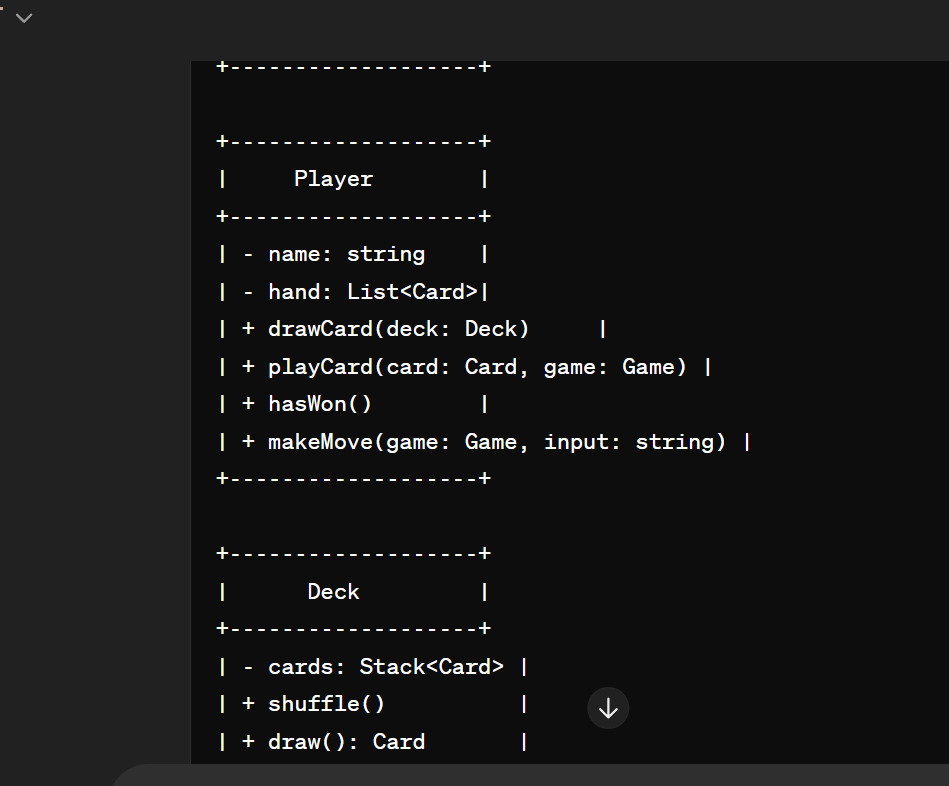
* **Untuk Apa:**
  + **handleInput** adalah metode yang digunakan untuk menerima dan memproses input dari pemain. Ini biasanya mencakup tindakan seperti memilih kartu untuk dimainkan atau menarik kartu dari dek.
* **Fungsi:**
  + Menerima input dari pemain, yang bisa berasal dari GUI, CLI, atau metode input lainnya.
  + Memproses input tersebut dan menjalankan aksi yang sesuai (misalnya, memanggil **playCard** atau **drawCard**).
  + Mengelola logika permainan berdasarkan aksi pemain, termasuk validasi dan pengaturan giliran.

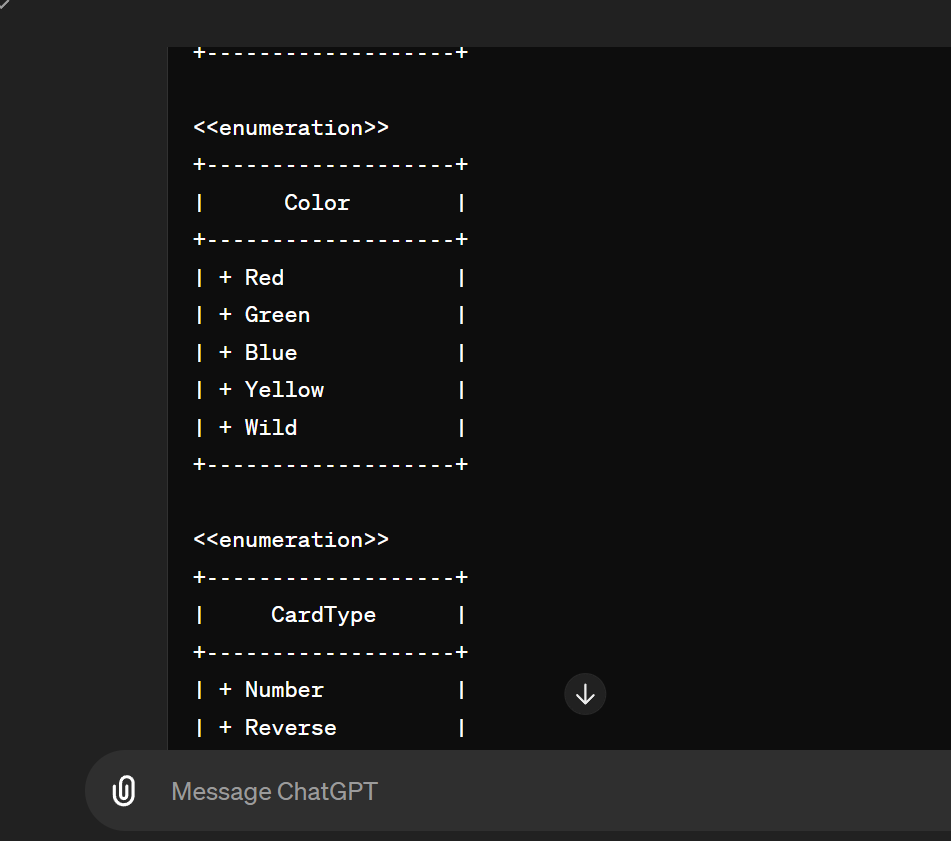


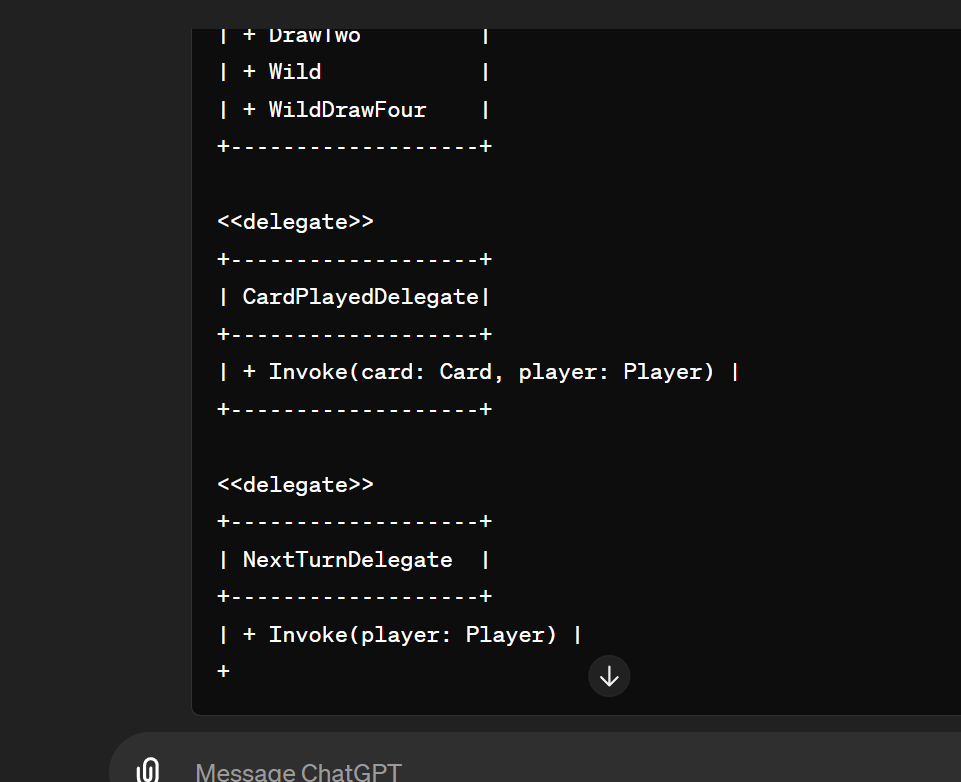










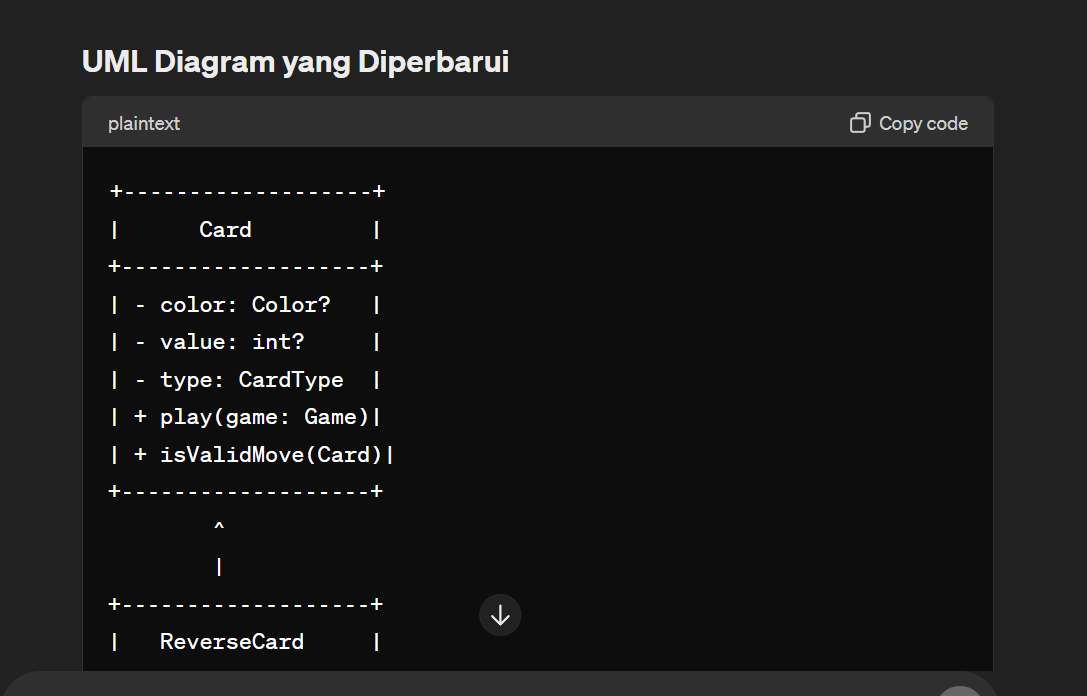


Ya, Anda benar. Mari kita perjelas perbedaan antara **MakeMove** di **GameController** dan **PlayCard** di **Player**:

1. **MakeMove di GameController**:
   * **MakeMove** digunakan untuk memproses langkah yang diambil oleh pemain. Ini bisa termasuk memvalidasi apakah langkah yang diambil oleh pemain sesuai dengan aturan permainan, seperti apakah kartu yang dilemparkan oleh pemain sesuai dengan kartu teratas di tumpukan buangan.
   * Jika langkah pemain valid, **GameController** kemudian akan menjalankan aksi yang sesuai, seperti memperbarui status permainan (misalnya, memindahkan kartu ke tumpukan buangan), memperbarui giliran pemain berikutnya, dan mungkin juga menangani efek tambahan dari kartu yang dimainkan.
   * Jika langkah pemain tidak valid, **GameController** akan memberi tahu pemain bahwa langkah tersebut tidak diterima.
2. **PlayCard di Player**:
   * **PlayCard** adalah metode yang dipanggil oleh pemain untuk memainkan kartu dari tangan mereka.
   * Pada tingkat pemain, ini adalah langkah pertama dalam proses memainkan kartu. **Player** memilih kartu dari tangan mereka yang ingin dimainkan dan memanggil **PlayCard** untuk melakukannya.
   * **PlayCard** di **Player** mungkin juga memiliki validasi sederhana, seperti memastikan bahwa kartu yang dipilih oleh pemain benar-benar ada di tangan mereka, tetapi validasi lebih lanjut biasanya dilakukan di **GameController** untuk memastikan kecocokan kartu dengan aturan permainan.

Jadi, **MakeMove** di **GameController** bertanggung jawab untuk mengawasi validitas langkah yang diambil oleh pemain secara keseluruhan, sedangkan **PlayCard** di **Player** lebih fokus pada langkah-langkah yang diambil oleh pemain secara individu saat mereka memainkan kartu dari tangan mereka.

DIPERBARUI







bagaimana dengan class numbercard dengan enumeration

ChatGPT

Karena kelas **NumberCard** menggunakan **Number** dari enumerasi sebagai salah satu atributnya, maka hubungan antara kelas **NumberCard** dan enumerasi **Number** adalah hubungan yang sangat erat. Dalam konteks ini, kita bisa menggunakan hubungan composition atau association untuk menunjukkan bahwa **NumberCard** menggunakan atau memiliki referensi ke nilai-nilai enumerasi **Number**.

Dalam UML, hubungan antara **NumberCard** dan **Number** bisa direpresentasikan sebagai berikut:

* **Association**: Association menunjukkan hubungan yang lebih longgar antara dua kelas. Dalam hal ini, **NumberCard** menggunakan **Number** sebagai salah satu atributnya, sehingga kita bisa menggunakan association untuk menunjukkan hubungan ini.