**BoardComponent**

**Zweck:** Zentrale Schnittstelle des Spielbretts, die das gesamte Spielgeschehen koordiniert.

* **Schlüsselmethoden und -statements:**
  + **initializeLayout:** Verantwortlich für die Strukturierung des Spielbretts. Nutzt das BorderLayout, um CardPanel und UserPanel zu platzieren, wodurch ein intuitives Layout für den Spieler entsteht.
  + **setupGameEngine:** Diese Methode ist entscheidend für die Initialisierung der GameEngine. Sie verknüpft die Spiellogik mit dem Spielbrett, indem sie sicherstellt, dass alle Aktionen des Spielers richtig verarbeitet und dargestellt werden.
  + **toggleHackerMode:** Eine Schlüsselkomponente für das interaktive Element des Spiels. Diese Methode aktiviert eine besondere Funktion, die das Spiel dynamischer und herausfordernder macht.

**Card**

**Zweck:** Repräsentiert eine einzelne Spielkarte mit zwei Zuständen (Vorder- und Rückseite).

* **Schlüsselmethoden und -statements:**
  + **flipCard:** Eine zentrale Methode, die das Umdrehen der Karte steuert. Sie ist entscheidend für das Kerngameplay, da das Aufdecken und Verdecken von Karten das Hauptziel des Spiels ist.
  + **isMatching:** Diese Methode ist essentiell für die Überprüfung von Kartenpaaren. Sie stellt die Grundlage für das Gewinnen des Spiels dar.

**CardPanel**

**Zweck:** Organisiert und präsentiert die Karten in einem visuell ansprechenden Gitterlayout.

* **Schlüsselmethoden und -statements:**
  + **createBoardLayout:** Diese Methode ist entscheidend für das visuelle Layout des Spiels. Sie passt das Gitterlayout an den gewählten Schwierigkeitsgrad an und beeinflusst somit direkt die Spielerfahrung.
  + **createCards:** Fundamental für das Gameplay, da sie für die Initialisierung und Zufallsverteilung der Karten zuständig ist. Dies trägt zur Spielbarkeit und Wiederspielbarkeit bei.

**CustomButtonModel**

**Zweck:** Bietet eine erweiterte Anpassungsfähigkeit für Buttons, die in der gesamten Anwendung verwendet werden.

* **Schlüsselmethoden und -statements:**
  + Die Klasse ist eine direkte Erweiterung von DefaultButtonModel und behält ihre Standardfunktionalität bei. Diese Klasse ist ein Beispiel für die Nutzung von Vererbung in Java, um bestehende Funktionen zu nutzen und bei Bedarf zu erweitern.

**ExitDialog**

**Zweck:** Ermöglicht dem Spieler, das Spiel zu beenden oder fortzusetzen.

* **Schlüsselmethoden und -statements:**
  + **initializeMessageLabel:** Diese Methode setzt die Nachricht, die dem Spieler präsentiert wird. Sie ist ein entscheidender Teil der Benutzeroberfläche, der dem Spieler klare Optionen bietet.
  + **configureButtonActions:** Eine kritische Methode, die festlegt, was geschieht, wenn der Spieler eine Wahl trifft. Sie verbindet die Benutzeroberfläche mit der Spiellogik.

**GameEngine**

**Zweck:** Kernklasse für die Verwaltung des Spielablaufs und der Zustände.

* **Schlüsselmethoden und -statements:**
  + **run:** Diese Methode ist das Herzstück der Spiellogik. Sie führt eine fortlaufende Schleife aus, die den Spielzustand überwacht und auf Veränderungen reagiert.
  + **handleNormalGameplay:** Verantwortlich für das Standard-Gameplay, außerhalb spezieller Modi wie dem Hacker-Modus. Diese Methode stellt die tägliche Spielroutine dar, in der der Spieler Karten aufdeckt und Paare sucht.

**GameResults**

**Zweck:** Zeigt die Ergebnisse nach dem Abschluss eines Spiels an.

* **Schlüsselmethoden und -statements:**
  + **formatTime:** Eine Methode, die die Spielzeit in ein benutzerfreundliches Format umwandelt. Sie ist wichtig für die Darstellung der Spielerleistung.

**GameTimer**

**Zweck:** Misst die Zeit, die der Spieler für das Spiel benötigt.

* **Schlüsselmethoden und -statements:**
  + **start/stop:** Methoden zum Steuern des Timers. Diese Methoden sind wesentlich, da sie die Grundlage für die Zeitmessung im Spiel bilden.
  + **updateTimer:** Aktualisiert regelmäßig die Zeit und kommuniziert sie an das UserPanel. Diese Methode ist zentral für die Darstellung der verstrichenen Spielzeit.

**JavamoryFrame**

**Zweck:** Hauptfenster der Anwendung, das als Container für alle Spielkomponenten dient.

* **Schlüsselmethoden und -statements:**
  + **initializeFrame:** Setzt die grundlegenden Eigenschaften des Fensters und lädt die Hauptkomponenten. Diese Methode ist entscheidend für die initiale Darstellung des Spiels.

**JavamoryGame**

**Zweck:** Startpunkt des Spiels, der das Hauptfenster initialisiert.

* **Schlüsselmethoden und -statements:**
  + **main:** Die Hauptmethode, die beim Start des Programms ausgeführt wird. Sie ist der Einstiegspunkt für das gesamte Spiel.

**LevelDialog**

**Zweck:** Erlaubt dem Spieler, den Schwierigkeitsgrad für ein neues Spiel auszuwählen.

* **Schlüsselmethoden und -statements:**
  + **initializeComponents:** Initialisiert die Komponenten des Dialogs. Diese Methode ist wichtig für die Bereitstellung einer Benutzeroberfläche, die dem Spieler ermöglicht, das Spiel nach seinen Wünschen zu konfigurieren.

**MenuButton**

**Zweck:** Bietet eine anpassbare Schaltfläche für das Menü.

* **Schlüsselmethoden und -statements:**
  + **createButtonBorder:** Eine Methode, die dem Button ein einzigartiges visuelles Erscheinungsbild verleiht. Sie ist ein Beispiel für die Anpassung von Standard-Swing-Komponenten, um die Benutzeroberfläche attraktiver zu gestalten.

**MenuC**

**Zweck:** Präsentiert das Hauptmenü des Spiels mit verschiedenen Optionen.

* **Schlüsselmethoden und -statements:**
  + **addMenuComponents:** Eine Schlüsselmethode, die das Menü mit funktionalen und interaktiven Schaltflächen ausstattet. Sie ist entscheidend für die Navigation und die Auswahl von Spieloptionen durch den Spieler.

**SelectLevelPanel**

**Zweck:** Bietet eine Benutzeroberfläche zur Auswahl des Schwierigkeitsgrades.

* **Schlüsselmethoden und -statements:**
  + **createButton:** Wichtig für die Erstellung von Schaltflächen, die es dem Spieler ermöglichen, den Schwierigkeitsgrad auszuwählen. Diese Methode beeinflusst direkt die Spielerfahrung, indem sie das Spiel an den Spieler anpasst.

**UserPanel**

**Zweck:** Zeigt Informationen wie Versuche und verstrichene Zeit an und bietet Spielsteuerungsoptionen.

* **Schlüsselmethoden und -statements:**
  + **createPanel:** Organisiert die Anzeige von Versuchen, Zeit und Steuerungsschaltflächen. Diese Methode ist zentral für die Informationsdarstellung und die Interaktion des Spielers mit dem Spiel.