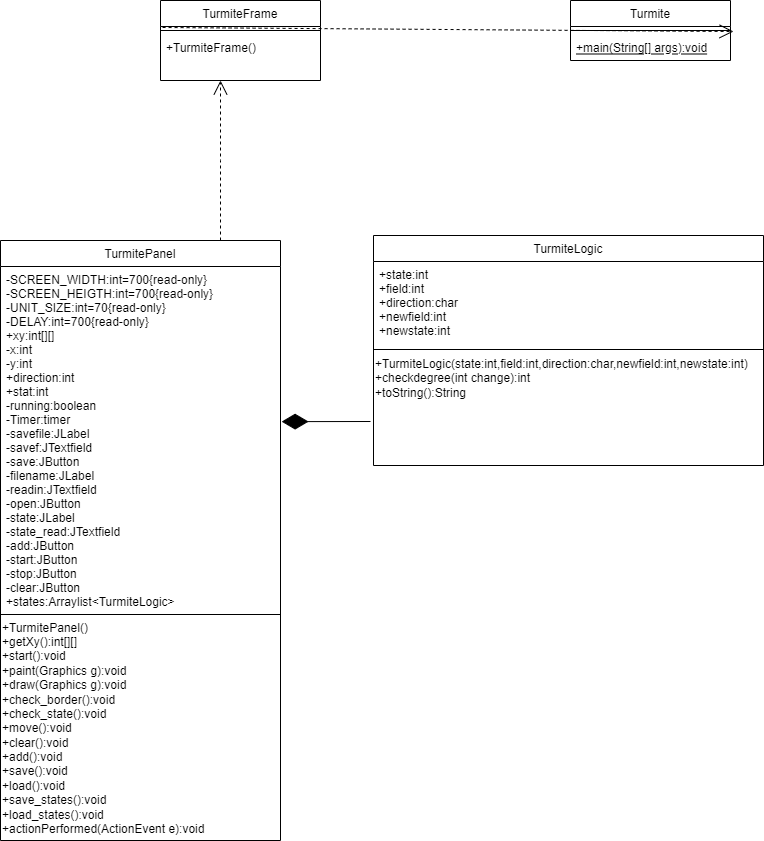
NHF Dokumentáció

**UML Táblázat:**



**Metódusok:**

TurmiteFrame:

Ennek az osztálynak csak egy konstruktora van amelyben létrehozom a JFrame-t, majd hozzá adom a JPanel-t.

**TurmitePanel:**

TurmitePanel():Az osztály konstruktora ahol létrehozom a gombokat és a területet ahol a program rajzolni fog. A létrehozott gomboknak is itt adom meg a funkcionalitásukat.

getxy(): A főmátrixnak az adatait tudom lekérdezni.

start(): Az animáció elindításáért felelős.

paint(Graphics g): A rajzfelület újra rajzolásáért felelős.

draw(Graphics g): A rajzfelületet és a mozgó elemeket jeleníti meg.

check\_border(): Ellenőrzi ha a mozgó ikon eléri a rajzfelület szélét. Ha igen leállítja az animációt.

check\_state():Ez a metódus olvassa ki az állapotokat tároló listából az aktuálisat majd ez alapján beállítja a paramétereket ami alapján tovább fog mozogni az ikon.

add():Új állapotot ad hozzá az állapotokat tároló listákhoz.

save():Elmenti a rajzot tároló mátrixot egy .txt fájlba.

load():Betölti a mátrixot tároló listából az adatokat majd kirajzolja az elmentet rajzot.

save\_states():Elmenti szerializálással az állapotokat tároló listát egy másik .txt fájlba.

load\_states():Beolvassa a .txt fájlból az állapotokat és eltárolja a listába őket. Ezzel a listában korábban eltárolt állapotokat is törli.

actionPerformed(ActionEvent e):Ez a függvény mozgatja a teljes kirajzolást ami a ’running’ boolean változó alapján működik.

TurmiteLogic:

TurmiteLogic(state:int,field:int,direction:char,newfield:int,newstate:int): A megadott állapotok alapján létrehoz egy változót. Ezeket használom az állapotok tárolására. Ezeket egy ArrayList-ben tárolom a TurmitePanel osztályban.

check\_degree(int change):Átadom neki a TurmitePanel direction változóját majd az épp aktuális irány alapján beállítja a helyes fordulást ügyelve arra hogy ne kapjunk vissza olyan irányt amit a program nem tud kezelni.

toString():Egyszerű toString metódus amit szerializálásnál használok.

**Felhasználói Könyv:**

A program elindításánál egy ablak jelenik meg aminek a nagyrészén baloldalt egy négyszetrácsos mező jelenik meg ahol a rajzolás fog történni. Ezen a részen mi semmit se tudunk csinálni csak a gép használja.   
Jobboldalt 3 szövegmező és 4 gomb jelenik meg:

* ’Clear’ Gomb:
  + A teljes rajzfelületet és az összes állapotot a listából törli a piros mutatót pedig visszaállítja középre. Ha ezt megnyomtuk akkor már nincs lehetőség a korábbi állapotok vissza hozására.
* ’Filename for save’ szöveghely:
  + Lehetőségünk van elmenteni a kirajzolt ábrát és a beírt állapotainkat. A szöveghelyre beírt név alapján fogja elmenteni a program ezeket melyeket a lejjebb leírt módszerrel tudunk vissza olvasni.
* ’Save’ Gomb:
  + A ’Filename for save’ szöveghelyre beírt szöveg alapján fogja elmenteni a programunkat.
* ’Filename’ szöveghely:
  + A korrábban elmentett fájlnak a nevét kell beírni. A visszaolvasással töröljük a korábbi állapotokat és a korábban elmentekkel töltjük fel a listánkat. Ezenfelül kirajzolja a korábban elmentet ábránkat.
* ’Open’ gomb:
  + A ’Filename’ szöveghelyre beírt szöveg alapján beolvassa a korábbi állapotokat és kirajzolja az ábárát.
* ’State(sta.-fie.-dir.-new fie.-new sta.)’ szöveghely:
  + A zárójelben megjelenített módon kell beírni az adatokat kötőjellel elválasztva:  
    (state(*állapot*)-field(*mező érték*)-direction(*irány*)-new field(*új mező érték*)-new state(*új állapot*))  
    Az állapotok és mezőértékek 0 és 1 közötti értékek lehetnek. Az irány pedig(L,R,N,U) kis vagy nagy betűvel.  
    Ezeket a gép eltárolja és figyelembe veszi az animációnál.
* ’Add’ gomb:
  + ’State(sta.-fie.-dir.-new fie.-new sta.)’ szöveghelyre beírt állapotot eltárolja a listában.
* ’Start’ gomb:
  + Elindítja a programot. Ezt megnyomva elindul az animáció és eltűnik az összes gomb majd megjelenik egy ’Stop’ gomb.
* ’Stop’ gomb:
  + Akkor jelenik meg ha elindítottuk az animációt. A gombot megnyomva leáll az animáció.