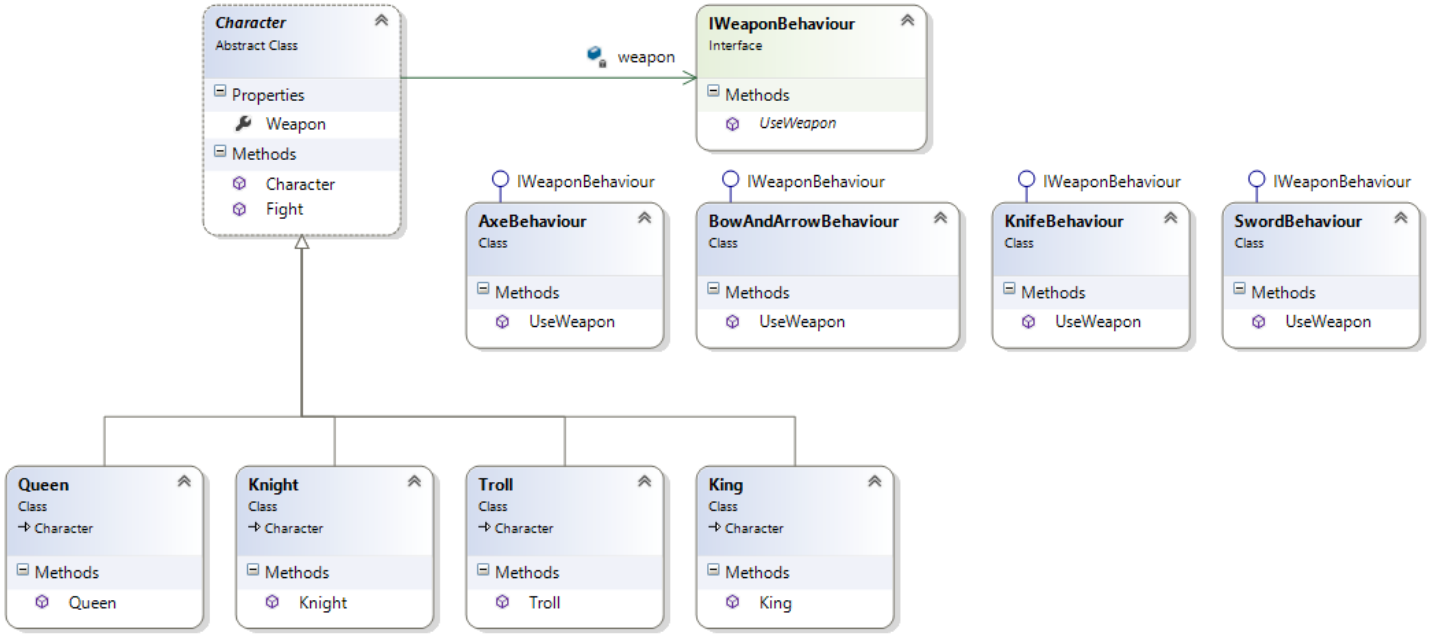
Opdracht 1 ('Strategy Pattern')

In een spel zitten de volgende karakters: Queen, Knight, Troll en King. Deze erven alle 4 van abstracte base class 'Character'. Elk karakter heeft een wapen waarmee hij kan vechten. Ontwerp hiervoor een interface IWeaponBehaviour en implementeer 4 verschillende soorten wapens: Axe, BowAndArrow, Knife en Sword die interface IWeaponBehaviour implementeren. Onderstaande class diagram laat de verschillende classes/interfaces zien.

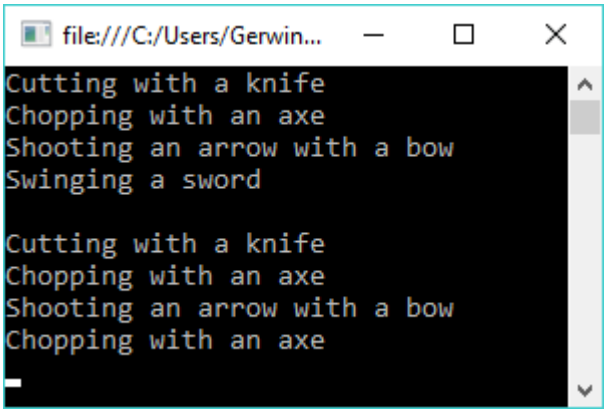


Elk karakter heeft een eigen default wapen, maar deze kan tijdens het spel gewijzigd worden (naar een ander wapen).

Implementeer bovenstaande classes/interfaces en gebruik het volgende Start-programma om het te testen:

```
void Start()
{
  List<Character> characters = new List<Character>();
  characters.Add(new Queen());
  characters.Add(new Troll());
  characters.Add(new King());
  characters.Add(new Knight());

  foreach (Character character in characters)
    character.Fight();
  Console.WriteLine();

  // change weapon of knight to axe
  characters[3].Weapon = new AxeBehaviour();

  foreach (Character character in characters)
    character.Fight();

  Console.ReadKey();
}
```

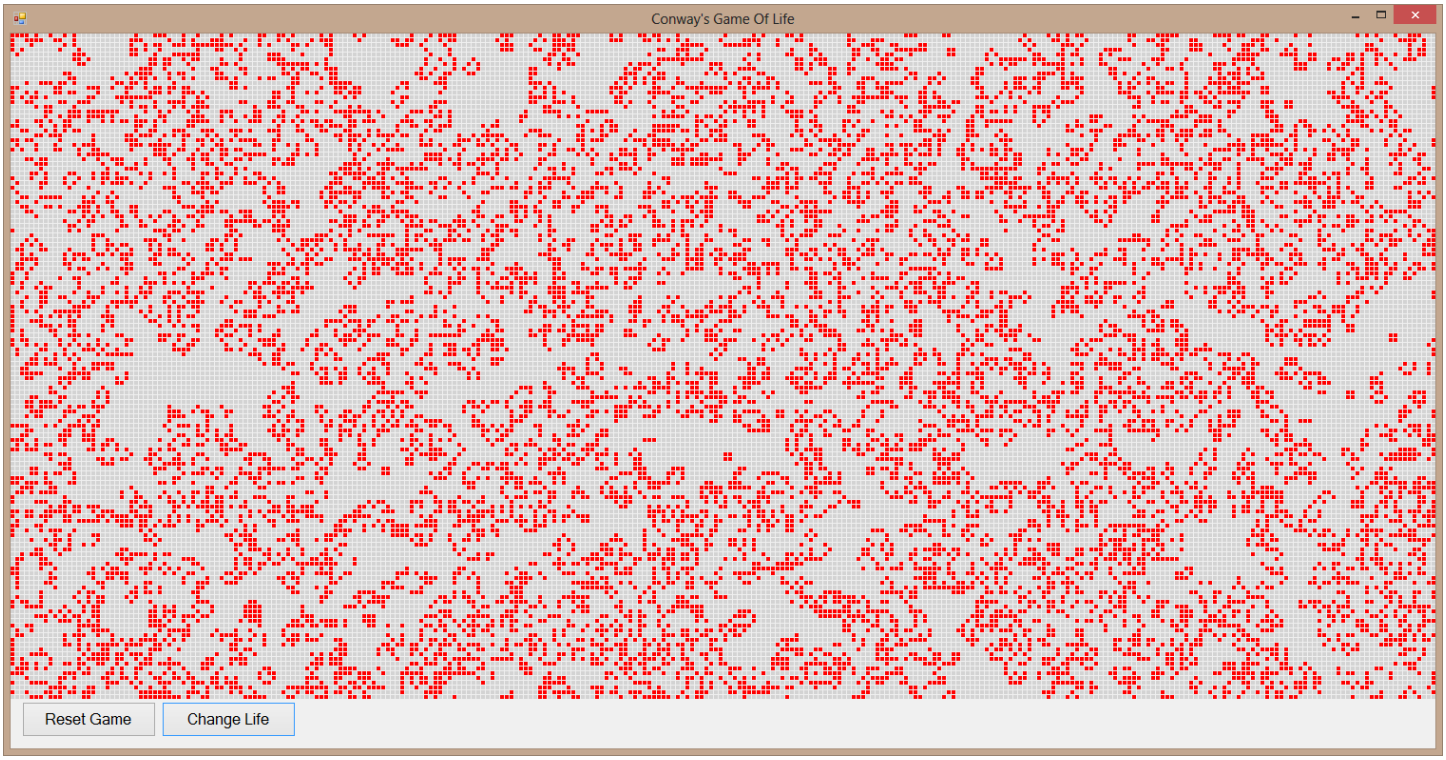
Deze code geeft de volgende uitvoer ➔



Opdracht 2 (‘Strategy Pattern’)

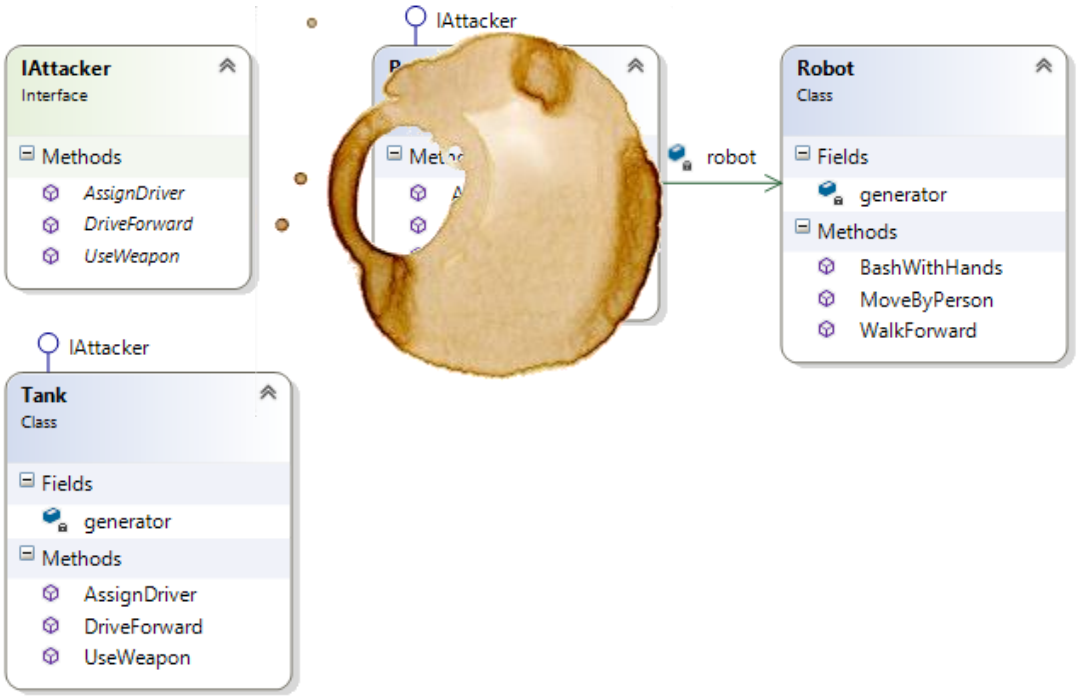
Op Moodle vind je onder ‘Week 4 opdrachten’ applicatie ‘Game of Life’ (dezelfde applicatie als van week 2). Deze applicatie werkt volgens de standaard leefregels van John Conway (B3/S23), zie uitvoer in de screenshot hieronder. Wijzig deze applicatie zodat er 2 varianten mogelijk zijn: een ‘standard life’ en een ‘high life’ variant (B36/S23, zie http://www.conwaylife.com/wiki/HighLife). Dezelfde opdracht dus als in week 2, maar implementeer nu de 2 varianten door de Strategy-pattern toe te passen op class `ConwayGameOfLife`, waardoor dynamisch het gedrag aangepast kan worden, dus zonder een nieuw hoofdobject te moeten maken. Methode ‘CellShouldLive’ (aangeroepen in methode ‘Evolve’) moet weer door 2 aparte classes (StandardLife en HighLife) geïmplementeerd worden, maar nu via een interface (‘ILifeBehaviour).

Voor meer informatie over Conway’s Game of Life, zie o.a. https://nl.wikipedia.org/wiki/Game_of_Life.



Opdracht 3 (‘Adapter Pattern’)

In een zeer kwaadaardig spel (niet geschikt voor onder 18 jaar…) worden verschillende ‘aanvallers’ gebruikt, waarmee vijanden verslagen kunnen worden. Zo is er onder andere een Tank, die interface IAttacker implementeert. In het spel moet nu een Robot worden toegevoegd die echter geen aanvaller is. De oplossing is om voor de robot een adapter te maken, zoals in onderstaande classdiagram is aangegeven (helaas is er koffie gemorst over deze classdiagram…).



Maak een applicatie waarin bovenstaande classes/interfaces zijn geïmplementeerd.
Run onderstaand hoofdprogramma dat de gegeven uitvoer toont.

```
void Start()
{
    // create a tank (and assign it to a driver)
    // ...

    // create a robot (and let it move by a person)
    // ...

    // create attackers list, and add tank and robot
    List<IAttacker> attackers = new List<IAttacker>();
    // ...

    // process all attackers
    foreach (IAttacker attacker in attackers)
    {
        attacker.DriveForward();
        attacker.UseWeapon();
    }
}
```