Opdrachten bij text adventure

Breidt de code uit door nieuwe classes te schrijven. De verhaallijn voor de speler is kort en krachtig. Schrijf hooguit enkele regels tekst per Room. Maximaal 10 Rooms, maar dat is geen harde limiet. Een beetje een verhaaltje is leuk natuurlijk, maar je schrijft code. Niet een boek of een lang verhaal.

Look!

Implementeer het command "look". Deze geeft opnieuw een beschrijving van de kamer (voor het geval de speler dat vergeten is).

Up/down

Implementeer "up" en "down" als mogelijke uitgangen voor bepaalde Rooms (als 'verdiepingen').

Player

Implementeer een Player. Een Player heeft een currentRoom.

```
public class Player
{
    public Room CurrentRoom { get; set; }

    public Player()
    {
        CurrentRoom = null;
    }
}
```

Verplaats currentRoom naar Player. De class Game mag alleen een parser en een player bevatten.

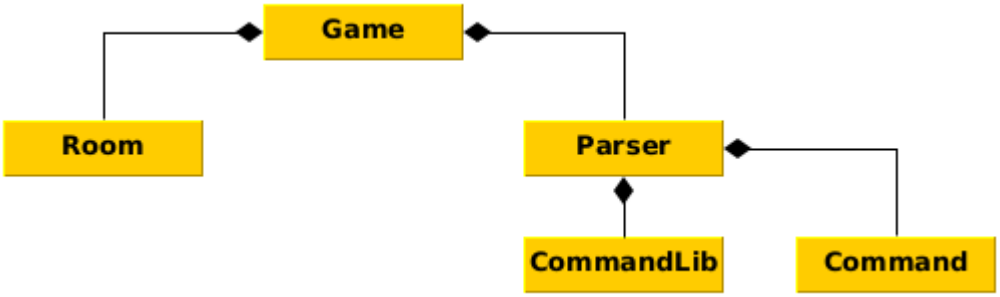
```
public class Game
{
    private Parser parser;
    private Player player;

    public Game ()
    {
        parser = new Parser();
        player = new Player();
        CreateRooms();
    }
}
```

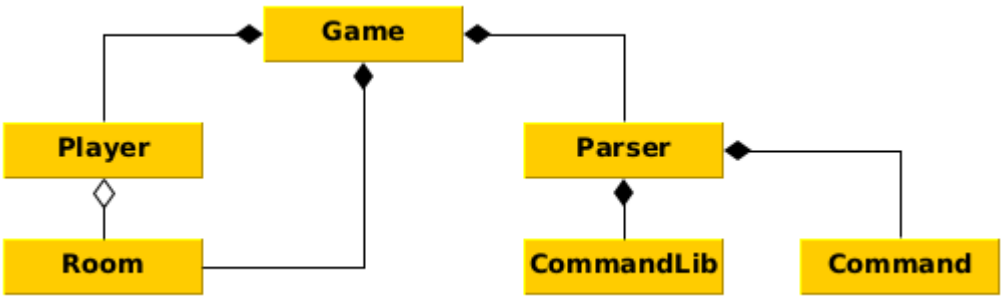
Vergeet niet om ook CreateRooms() aan te passen:

```
private void CreateRooms()
{
    // ...
    player.CurrentRoom = outside;
}
```

Oude situatie:



Nieuwe situatie:



Player status

Implementeer field health in Player.

```
public class Player
{
    private int health;

    public Player()
    {
        health = 100;
    }
}
```

Implementeer de volgende methods in Player:

```
Damage(amount) { /*...*/ }
Heal(amount) { /*...*/ }
IsAlive() { /*...*/ }
```

Van welk type variabele zijn de argumenten, en van welk type zijn de returnwaarden?

Gewond...

Elke keer als de speler naar een andere kamer gaat, verliest deze wat health (de speler is gewond, en verliest bloed bij het verplaatsen). Implementeer dit.

Uiteraard zorg je er voor, dat het spel is afgelopen als de speler "dood" is, of het spel heeft verloren.

Implementeer (eventueel later) ook dat de speler de adventure kan winnen, waarna het spel eindigt.

Items

In een Room liggen mogelijk Items (die een speler op kan pakken). Implementeer de class Item.

```
public class Item
{
    public int Weight { get; }
    public string Description { get; }

    public Item(int weight, string description)
    {
        Weight = weight;
        Description = description;
    }
}
```

Inventory

Implementeer de class Inventory. De class heeft de naam Inventory, en bevat een collection met Items. Welk type collection is het meest geschikt? (List, Dictionary, Stack, Queue, ...)

Is het ook handig om een korte Name in de class Item bij te houden? Is het nodig?

In een inventory kunnen Items worden gestopt. De speler kan maximaal xx kilo bij zich dragen. Implementeer ook de controle op het maximale gewicht van een Inventory.

```
public class Inventory
{
    private int maxWeight;
    private Collection<???> items;

    public Inventory(int maxWeight)
    {
        this.maxWeight = maxWeight;
        this.items = new Collection<???>();
    }

    public bool Put(string itemName, Item item)
    {
        // check the Weight of the Item!
        // put Item in the items Collection
        // return true/false for success/failure

        return false;
    }

    public Item Get(string itemName)
    {
        // find Item in items Collection
        // remove Item from items Collection if found
        // return Item

        return null;
    }
}
```

Inventory voor Room

Een Room heeft een Inventory. Laten we de Inventory voor de Room conceptueel een "chest" noemen. Een Player krijgt later ook een Inventory. Tussen Rooms en Player kunnen Items uitgewisseld worden.

Je kunt voor de Inventory van de Room een getter maken. Iedereen die een Room binnen loopt, mag zien welke Items er liggen. Een setter is niet nodig, niemand hoeft een complete chest te vervangen.

```
public class Room
{
    // field
    private Inventory chest;

    // property
    public Inventory Chest
    {
        get { return chest; }
    }

    public Room()
    {
        // a Room can handle a big Inventory.
        chest = new Inventory(999999);
    }
}
```

Inventory voor Player

In Player implementeer je de methods TakeFromChest() en DropToChest(). Hier handel je de uitwisseling van Items en communicatie met de speler af ("Item is not in Room", "Item doesn't fit in your inventory", of "You don't have that Item.").

De Inventory van de Player is private. Niemand mag daar zomaar bij.

```
public class Player
{
    private Inventory inventory;

    public Player()
    {
        // 25kg is pretty heavy to carry around all day.
        inventory = new Inventory(25);
    }

    public bool TakeFromChest(string itemName)
    {
        // remove the Item from the Room
        // put it in your inventory
        // inspect returned values
        // communicate to the user what's happening
        // return true/false for success/failure

        return false;
    }

    public bool DropToChest(string itemName)
    {
        // remove Item from your inventory.
        // Add the Item to the Room
        // inspect returned values
        // communicate to the user what's happening
        // return true/false for success/failure

        return false;
    }
}
```

Hoe zorg je ervoor dat de speler van het spel kan zien welke Items er in de Room liggen? En welke Items een Player in de Inventory heeft? Implementeer dit.

Take/drop

Nu kun je eindelijk het commando "take" en "drop" implementeren!

```
public class Game
{
    private void Take(Command command)
    {

    }

    private void Drop(Command command)
    {

    }
}
```

Use an Item

Hoe gebruikt de Player een Item eigenlijk?

```
public class Player
{
    public string Use(string itemName)
    {
        // TODO implement
    }
}
```

Inmiddels de hoogste tijd om een "thirdWord" aan Command toe te voegen. Je wilt immers een volgend Command in kunnen tikken:

```
use key east
```

of, als je meerdere wapens hebt:

```
attack guard axe
```

Implementeer dit.

Endgame

Dit zou een geschikt moment zijn om een gezonde speler gewond te laten raken. Gebruikt de speler een 'bad Item' bijvoorbeeld? Liggen er medkits in de Rooms?

Misschien worden bepaalde Rooms bewaakt door een guard die de speler moet bevechten? Of zijn sommige "Rooms" niet toegankelijk (locked), en heeft de speler hier een sleutel voor nodig?

Probeer dit te implementeren.