## Movie World

Create the following project structure



### Class Customer

The **Customer class** should receive the following parameters upon initialization: **name: str**, **age: int**, **id: int**. Each customer should also have an instance **attribute** called **rented\_dvds** (list with **DVD instances**; **empty** upon **initialization**).

Implement the **\_\_repr\_\_** method, so it **returns** the following string: **"{id}: {name} of age {age} has {count\_rented\_dvds} rented DVD's ({dvd\_names joined by comma and space})"**

### Class DVD

The **DVD class** should receive the following parameters upon initialization: **name: str**, **id: int**, **creation\_year: int**, **creation\_month: str**, **age\_restriction: int**. Each DVD should also have an **attribute** called **is\_rented** (**False** by default)

Create method called **from\_date(id: int, name: str, date: str, age\_restriction: int)** – it should create a **new instance** using the provided data. The **date** will be in format **"day.month.year" - all of them numbers.**

Implement the **\_\_repr\_\_** method so it returns the following string: **"{id}: {name} ({creation\_month} {creation\_year}) has age restriction {age\_restriction}. Status: {rented/not rented}"**

### Class MovieWorld

The **MovieWorld** class should receive **one parameter** upon initialization: **name: str**. Each **MovieWorld** instance should also have **2 more attributes**: **customers** (**list** of Customer **objects**, **empty** upon initialization), **dvds** (**list** of DVD **objects**, **empty** upon initialization). The class should also have the following **methods**:

* **dvd\_capacity()** –returns **15** - the **dvd capacity** of a movie world
* **customer\_capacity()** – returns **10** - the **customer capacity** of a movie world
* **add\_customer(customer: Customer)** – add the customer if capacity not exceeded
* **add\_dvd(dvd: DVD)** – add the dvd if capacity not exceeded
* **rent\_dvd(customer\_id: int, dvd\_id: int)**
  + If the customer has **already rented** that dvd return **"{customer\_name} has already rented {dvd\_name}"**
  + If the dvd **is rented by someone else**, return **"DVD is already rented"**
  + If the customer is **not allowed** to rent the DVD, return **"{customer\_name} should be at least {dvd\_age\_restriction} to rent this movie"**
  + Otherwise, the rent is **successful** (the dvd is rented and its added to the customer dvds). Return **"{customer\_name} has successfully rented {dvd\_name}"**
* **return\_dvd(customer\_id, dvd\_id)** – if the dvd is in **the customer**, he/she should **return it** and the method should return the message **"{customer\_name} has successfully returned {dvd\_name}"**. Otherwise return **"{customer\_name} does not have that DVD"**
* **\_\_repr\_\_()** – return the **string representation** of **each customer** and **each dvd** on separate lines

### Examples

|  |
| --- |
| **Test Code** |
| from project.customer import Customer  from project.dvd import DVD  from project.movie\_world import MovieWorld  c1 = Customer("John", 16, 1)  c2 = Customer("Anna", 55, 2)  d1 = DVD("Black Widow", 1, 2020, "April", 18)  d2 = DVD.from\_date(2, "The Croods 2", "23.12.2020", 3)  movie\_world = MovieWorld("The Best Movie Shop")  movie\_world.add\_customer(c1)  movie\_world.add\_customer(c2)  movie\_world.add\_dvd(d1)  movie\_world.add\_dvd(d2)  print(movie\_world.rent\_dvd(1, 1))  print(movie\_world.rent\_dvd(2, 1))  print(movie\_world.rent\_dvd(1, 2))  print(movie\_world) |
| **Output** |
| John should be at least 18 to rent this movie  Anna has successfully rented Black Widow  John has successfully rented The Croods 2  1: John of age 16 has 1 rented DVD's (The Croods 2)  2: Anna of age 55 has 1 rented DVD's (Black Widow)  1: Black Widow (April 2020) has age restriction 18. Status: rented  2: The Croods 2 (December 2020) has age restriction 3. Status: rented |