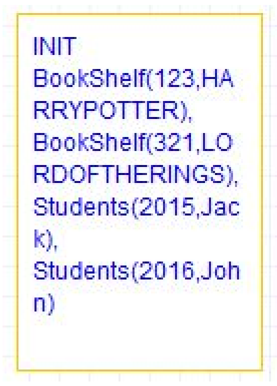
The Following document explains how we used Mista components to achieve the exercise. So, it is redundant to explain aspects that we are not going to use. For more information please go to the documents folder inside the "MISTA1.0.zip"

Steps to create a project:

1. Unzip "MISTA1.0.zip"
2. Go to MISTA1.0 -----> MISTA1.0 -----> MISTA.jar
3. Double-click MISTA.jar file
4. Save the MISTA project to the "test" folder of the project (Library/src/test/)
    a. Generated tests will be in the same directory as the MISTA project file.

MISTA Components:

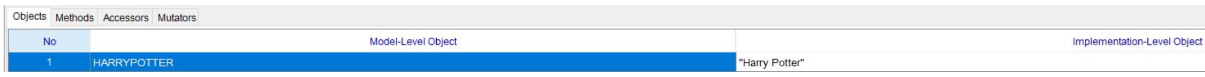
- Places ○ : are where data is held in the program. These hold tokens, which represent objects. Tokens move between places when transitions ▯ are triggered.
- Transitions ▯: These represent Events(i.e. Functions in java terminology).
- Arcs ↖ :connect between a Place and a Transition, and never between two of the same type. Additionally, they allow you to give the transition a direction.
- Annotations: We used annotations in this exercise to create initial Tokens (i.e. objects in java terminology) in the place called "Students".



```
INIT
BookShelf(123,HA
RRYPOTTER),
BookShelf(321,LO
RDOFTHERINGS),
Students(2015,Jac
k),
Students(2016,Joh
n)
```

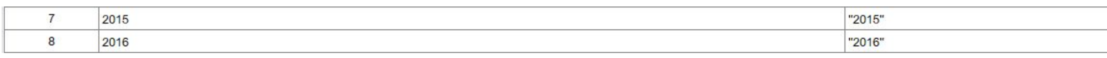
MISTA Terminologies:

- Objects:
    - Necessary for correct java test generation.
    - If you give a token a value such as HARRY POTTER, you should tell MISTA that when it translates to java it should be in double quotes “Harry Potter”



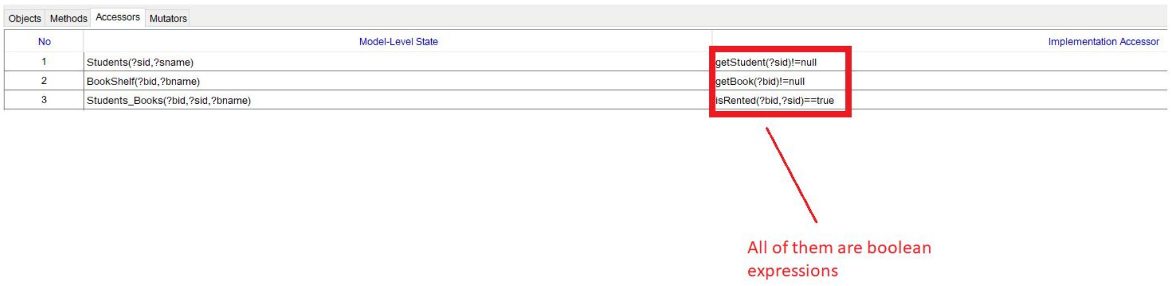
No	Model-Level Object	Implementation-Level Object
1	HARRYPOTTER	"Harry Potter"

    - If you have an integer in the token’s data in MISTA and you don’t want to change its type when the tests are translated to Java then you don’t have to do anything.
        - But, in our case, the Student ID is defined as a string in the java code.So, we had to do the following:



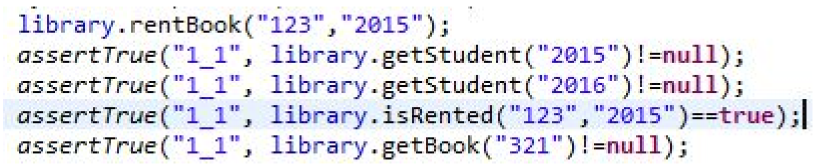
7	2015	"2015"
8	2016	"2016"

- Methods:
    - Necessary for correct java test generation.
    - We used it to map the Transition ▯ in MISTA “rentBook(?bname,?sname)” to a function in the java project “rentBook(?bname,?sname)” for example.
        - In our case both have the same name, but it could be different.
- Accessors:
    - Where places ○ are mapped to boolean expressions.



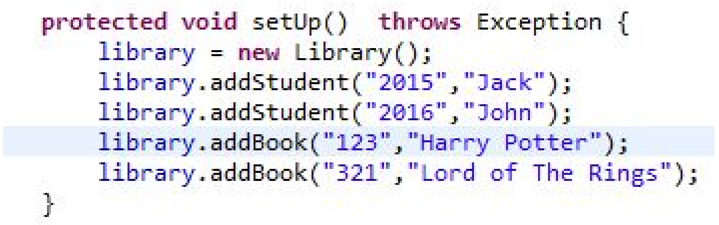
No	Model-Level State	Implementation Accessor
1	Students(?sid,?sname)	getStudent(?sid)!=null
2	BookShelf(?bid,?bname)	getBook(?bid)!=null
3	Students_Books(?bid,?sid,?bname)	isRented(?bid,?sid)==true

    - Every time a transition like “rentBook(bid,sid)” is triggered, all places connected to this Transition ▯ have their corresponding boolean expressions checked.
    - Necessary for correct java test generation.



```
library.rentBook("123","2015");
assertTrue("1_1", library.getStudent("2015")!=null);
assertTrue("1_1", library.getStudent("2016")!=null);
assertTrue("1_1", library.isRented("123","2015")==true);
assertTrue("1_1", library.getBook("321")!=null);
```

- Mutators:
    - Necessary for correct java test generation.
    - where it is used to set up the initial data for testing (Mutators are used by the “INIT” annotation in MISTA).



```
protected void setUp()  throws Exception {
    library = new Library();
    library.addStudent("2015","Jack");
    library.addStudent("2016","John");
    library.addBook("123","Harry Potter");
    library.addBook("321","Lord of The Rings");
}
```

Exercises:

- As you may have seen we have a library application written in Java.
- We have 2 exercises
    - Exercise 1: Model the rentBook function in MISTA and then generate test code and run the test code.
        - Don’t worry this whole process is shown in the video and you have to follow the steps.
    - Exercise 2: add new components to the model to test the function of returning a book to the library.
        - Once you finish modelling you should try and generate the test code and then run it using Junit in eclipse.
            - If no errors exist then you have finished!

Note: We will be happy to help you.