|  |  |  |
| --- | --- | --- |
| Book | | |
| ID | **Method** | **Test Description** |
| 01A | \_\_init\_\_ | Creates a Book object if the type of the attributes **book\_id**, **book\_title**, **author**, **published\_year** and **edition** are valid |
| 01B | \_\_init\_\_ | Book\_id length out of range:   * Less than 3 * Greater than 7   Published\_year out of range:   * Less than 1920 * Greater than 2020   edition:   * less than 1   invalid type of **book\_id** (!=str), **book\_title**(!=str), **author**(!=str), **published\_year**(!=int), **edition**(!=int) |
| 02A | get\_book\_id | Gets the valid/not none book id from the object in setUp method |
| 03A | get\_book\_title | Gets the valid/not none book title from the object in setUp method |
| 04A | get\_author | Gets the valid/not none book author from the object in setUp method |
| 05A | get\_published\_year | Gets the valid/not none book published year from the object in setUp method |
| 06A | get\_edition | Gets the valid/not none book edition from the object in setUp method |
| 07A | get\_availability\_status | Gets the valid/not none book availability status in the library from the object in setUp method |
| 08A | get\_type | Gets the valid/not none book type (ebook/textbook) from the object in setUp method |
| 09A | display\_info | Raises NotImplementedError |

|  |  |  |
| --- | --- | --- |
| eBook | | |
| ID | **Method** | **Test Description** |
| 01A | \_\_init\_\_ | Valid **platform** and **book\_category** and the book type is **ebook 🡪** Creates a **ebook** object |
| 01B | \_\_init\_\_ | platform not in the list  book\_category not in the list  Invalid type of platform value(!=str) and book\_category(!= str) |
| 02A | get\_book\_type | Gets the valid/not none book type (ebook) value from the object in setUp method |
| 03A | get\_book\_platform | Gets the valid/not none book type (ebook) value from the object in setUp method |
| 04A | get\_available\_platforms | Gets the valid/not none book platform value from the object in setUp method |
| 05A | get\_book\_genre | Gets the valid/not none book genre (ebook) value from the object in setUp method |
| 06A | suffix | Gets the valid ebook edition suffix from the object in setUp method |
| 07A | display\_info | Gets the valid/not none ebook availability status in the library from the object in setUp method |

|  |  |  |
| --- | --- | --- |
| Textbook | | |
| ID | **Method** | **Test Description** |
| 01A | \_\_init\_\_ | Valid **cover\_type**, **book\_subject** and the type is **textbook 🡪** Creates a **textbook** object |
| 01B | \_\_init\_\_ | **cover\_type** and **book\_subject** not in the list specified  Invalid type of cover type (!=str) and book subject (!) |
| 02A | get\_book\_type | Gets the valid/not none book type value from the object in setUp method |
| 03A | get\_book\_subject | Gets the valid/not none book subject value from the object in setUp method |
| 04A | get\_cover\_type | Gets the valid/not none book cover type value from the object in setUp method |
| 05A | get\_available\_cover\_types | Gets the valid/not none book cover types available in the library value from the object in setUp method |
| 06A | suffix | Gets the valid textbook edition suffix from the object in setUp method |
| 07A | display\_information | Gets the valid/not none textbook availability status in the library from the object in setUp method |

* How does your design implement the four pillars of OOP (abstraction, encapsulation, inheritance and composition, and polymorphism)?
  + Abstraction: we have two abstract methods of **get\_book\_type** and **display\_info**
  + Encapsulation**:**  as we have private attributes, they are encapsulated in such a way that they cannot be called or accessed directly. By creating objects, we are encapsulating.
  + Inheritance & composition: ebook and textbook classes have inherited common attributes and methods of book class.
  + Library\_stat object is an example of composition which means that library stats object is created in LibraryManager class.
  + Polymorphism: two methods of **get\_book\_type** and **display\_info** are examples of abstract methods which we also use them in ebook and textbook classes. They behave differently based on the class they are in.
* Why are your classes good abstractions (i.e., models) of the real-world entities they represent?

Because they have real attributes and methods of real entity (library) thus they can behave as a real-world entity. In another world, they are encapsulated in such a way that only the needed and relevant details of the entity is shown to public interface.