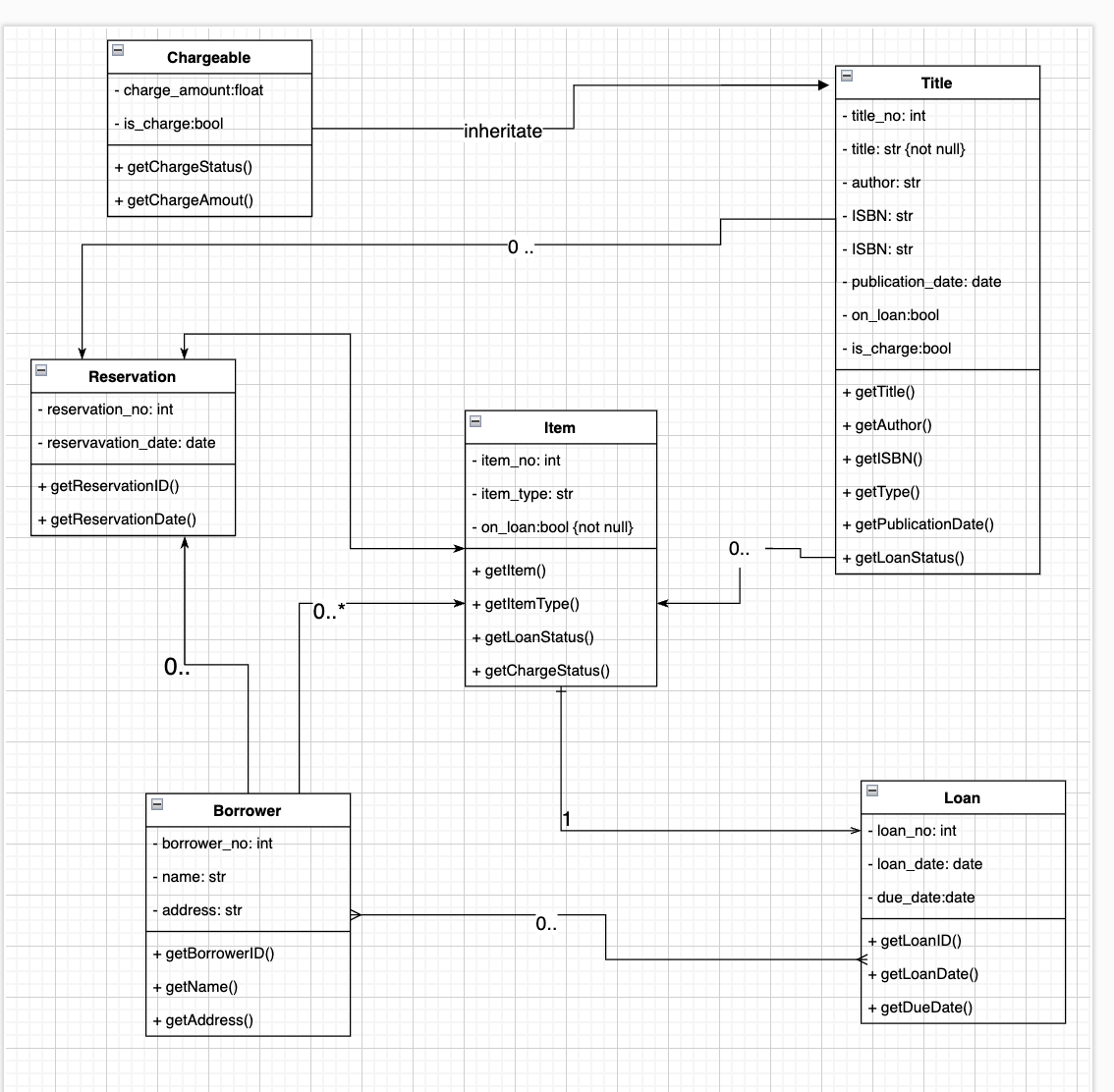
**1. Design Classes on a class diagram.**



Inheritance is an essential technique in object-oriented programming that allows classes to share attributes and behaviors. It can reduce code duplication and make the code more maintainable by organizing related classes into a hierarchy. The Chargeable class can inherit from the Title class using inheritance. This means that the Chargeable class inherits all the attributes and methods from the Title class, and can also have its own attributes and methods.

Here is a list of the associations, multiplicities, and navigabilities

* Reservation and Item:
  + Association name: involves
  + Multiplicity: 1 Reservation involves 1 Item
  + Navigability: Reservation can access Item, but Item cannot access Reservation
* Title and Item:
  + Association name: has
  + Multiplicity: 1 Title has many Items
  + Navigability: from Title to Item
* Item and Loan:
  + Association name: checked out by
  + Multiplicity: 1 Item is checked out by 1 Loan
  + Navigability: from Item to Loan
* Borrower and Loan:
  + Association name: has
  + Multiplicity: A Borrower class can have many Loan classes and a Loan class can belong to many Borrowers.
  + Navigability: Borrower can access Loan, but Loan cannot access Borrower
* Borrower and Item:
  + Association name: checkedOut
  + Multiplicity: 1 Borrower checks out many Items
  + Navigability: Borrower can access Item, but Item cannot access Borrower
* Borrower and Reservation:
  + Association name: has
  + Multiplicity: 1 Borrower makes many Reservations
  + Navigability: Borrower can access Reservation, but Reservation cannot access Borrower
* Reservation and Item:
  + Association name: has
  + Multiplicity: An Item class can have many Reservation classes, but a Reservation class can only belong to one Item class.
  + Navigability: Reservation can access Item, but Item cannot access Reservation
* Title and Reservation:
  + Association name: reserve
  + Multiplicity: 1 Title is reserved for many Reservations
  + Navigability: from Title to Reservation

2. Explain the outcomes

Association between classes is represented by a solid line connecting two classes, with an optional arrowhead indicating navigability (in which direction the connection is navigable). The association suggests that instances of one class are related to instances of another class.

Aggregation is an association where an instance of one class can contain models of another class. A diamond shape on the end of the line near the containing class represents this on a class diagram. For example, in the library system, the Title class might contain multiple Item instances representing the different copies of a book.

Composition is a more restrictive form of aggregation, where the contained cases cannot exist independently of the containing instance. A filled diamond shape on the end of the line near the containing class represents this on a class diagram.

Inheritance is represented by a solid line with an arrowhead pointing from the subclass (or derived class) to the superclass (or base class). Inheritance indicates that the subclass is a specialized version of the superclass, and that it inherits all of the attributes and methods of the superclass.

To determine if a class "has a" attribute for another class, consider if the attribute is a part of the class or is related to the class somehow. If the attribute is a part of the class, it can be listed as an attribute of the class in the class diagram. If the attribute is related to the class, it might be represented as an association on the class diagram. For example, if a Borrower has a name, the name attribute can be listed as part of the Borrower class in the class diagram. If a Borrower also has a loan record, the loan record can be represented as an association between the Borrower and Loan classes.

Generalization and specialization techniques create a hierarchy of classes that share common attributes and behaviors. Generalization involves creating a superclass that encompasses several subclasses' common attributes and behaviors. Specialization involves creating subclasses that inherit from the superclass and specializing it by adding additional attributes or behaviors.