Design a class hierarchy for the items in a libary. Put these classes together in a package that is separate from the package that `main` is in. The main method will represent a commond-line interface to a library system that will be implemented throughout the rest of this lab.

Some suggestions: * `main` is in a `com.tts.oop` package. * The library classes are in the `com.tts.oop.model` package. * `LibraryItem` is a base class, two immediate derived classes are `Book` and `Periodical`. Derived from `Book` are `ReferenceBook` and `GeneralBook`, and from `Periodical` are `Magazine` and `NewsPaper`. * What other types can derive from `LibraryItem`? Audio? Video? And sub-types of them could be? * What is common to all `LibraryItems`? To `Book`? Make sure the properties and methods are in their appropriate place, and overridden where necessary.

- A common line of thinking is this: If you find yourself defining the same property in many derived classes, this indicates the property *could* be defined in a base class. The same with methods - if there is a similar method definition occurring in many child classes, or a definition that is only *slightly* different, this may be a base class method that can be overridden when necessary.

Write a class for a `CardHolder` to represent someone who joined the library and can check out items, reserve items, etc.

Interfaces

Some items may be reservable, and some may be loanable. Create an interface for each of these and implement them in classes which fit the appropriate type.

Relationships

Create classes for Author and Publishing Company. Code in relationships into each of the library items. How will you handle that an item can have multiple authors?

Generics

How have generics been used so far? Write a custom generic collection called `OverdueList` to manage a list of overdue items. Create separate members in the `CardHolder` for a list of books, a list of periodicals, and a list of multimedia items that are overdue.

Refactor the `Loanable` interface to include a late charge. Also, create an enumeration for how frequently late charges are added - Daily, Weekly, Bi-Weekly, Monthly, etc.

Patterns

Read the following articles on design patterns: *Design Patterns in java * Factory Design Pattern * Publish-Subscribe

Create a factory for Library Items. How will you determine which item to create? A lookup? Enum? String?

Think about how you would implement a reservation notification system. This sub-system would let users sign up for an alert if an item they were interested in was checked out and then became available.