Lab 10 -A

You are tasked with developing a console-based library management system that will allow users to manage books, DVDs, and CDs. The system should have the following functionalities:

- ➔ Users can add a new item to the library's collection of books, DVDs, or CDs.
- ➔ Users can remove an existing item from the library's collection of books, DVDs, or CDs.
- ➔ Users can search for an item in the library's collection by title, author, or category.
- ➔ Users can borrow and return items from the library's collection.
- ➔ Users can view a list of all available items in the library's collection.

Your task is to develop a console application that implements the above functionalities using OOP concepts such as inheritance, polymorphism, generics, and custom exception classes.

Requirements

Your console application should have the following classes:

1) Item - This is an abstract class that represents a generic item in the library's collection. It should have the following properties:
   a) id (string) - a unique identifier for the item.
   b) title (string) - the title of the item.
   c) author (string) - the author of the item.
   d) category (string) - the category of the item (book, dvd, or cd).
   e) isAvai1ab1e (boolean) - whether the item is currently available for borrowing.

   It should also have the following methods:
   a) displayDetails () - a method that displays the details of the item (id, title, author, category, and availability).
   b) borrowItem() - a method that sets the isAvaiIabIe property to false when an item is borrowed.

c) returnItem() - a method that sets the isAvailable property to true when an item is returned.

2) Book - This is a subclass of the Item class that represents a book in the library's collection. It should have the following additional properties:
   a) isbn (string) - the ISBN number of the book.
   b) numPages (int) - the number of pages in the book.

   It should also have a constructor that takes in the id, title, author, category, isbn, and numPages as parameters and initializes the corresponding properties.
3) DVD - This is a subclass of the Item class that represents a DVD in the library's collection. It should have the following additional properties:
   a) director (string) - the director of the DVD.
   b) length (int) - the length of the DVD in minutes.

   It should also have a constructor that takes in the id, title, author, category, director, and length as parameters and initializes the corresponding properties.
4) CD - This is a subclass of the Item class that represents a CD in the library's collection. It should have the following additional properties:
   a) artist (string) - the artist of the CD.
   b) numTracks (int) - the number of tracks on the CD.

   It should also have a constructor that takes in the id, title, author, category, artist, and numTracks as parameters and initializes the corresponding properties.
5) Library - This class represents the library and its collection of items. It should have the following properties:
   a) items (List<Item>) - a list of all the items in the library's collection. It should also have the following methods:
   b) addItem(item: Item) - a method that adds a new item to the library's collection.
   c) removeItem(item: Item) - a method that removes an existing item from the library' s collection.
   d) search Items (query: string) - a method that searches for items in the library's collection by title, author, or category.
   e) borrowItem(itemId : string) - a method that borrows an item from the library's collection by setting its isAvailable property to false.
   f) returnItem(itemId . string) - a method that returns a borrowed item to the library's collection by setting its isAvailable property to true.

g) displayAvai1ableItems() - a method that displays a list of all available items in the library ' s collection.

6) LibraryException - This is a custom exception class that should be thrown whenever an operation on the library's collection of items fails.

Your console application should have a menu that allows users to perform the above functionalities by interacting with objects of the Library class. The menu should be implemented using a loop that keeps prompting the user to input a choice until they choose to exit the application. You should also use generics wherever applicable to make your code more flexible and reusable.