Library Case - Exercise

1. Write a simple class – Book with the following fields/attributes:

* id : int
* name : String
* price : double
* add a constructor that takes and populates all fields
* implement getters/setters
* override equals(Object o) to return true only is compared with other Book instance that has the same values in all fields
* override toString() to return a string representation as follows:

Book: <name> - <price>

2. Extend Book with a new class named – BestSeller with the following additional fields:

* summary : String
* worldCopies : int
* complete getters/setters and contractor as needed
* update equals(Object o) to return true only is compared with other Book instance that has the same values in all fields
* update toString() to return a string representation as follows:

Best Seller: <name> - <price> Summary: <summary> , <worldCopies> Copies Sold !

3. Create a Storage class. This class holds two arrays which are in synch:

* books : Book[]
* inStock : int[]

books holds all Books in stock, while inStock holds the amount of each. For examples:

if the library has 5 copies of "The Code Book" and it is the first book added to the stock,

books[0] will hold the Book instance of "The Code Book" and inStock[0] will hold the value 5.

Add the following methods :

* constructor that instantiates the arrays with initial size of 6
* addBook(Book newBook, int amount) – adds a books to the books array and adds the amount to inStock. Use System.arraycopy() when needed
* rentBook(Book book) : String - reduces inStock array and returns book name as approval or null if no books in stock (in stock = 0)
* returnBook(Book book) – updates book in stock value
* getInStock(Book book) : int – returns the current amount of the given book in stock

4. Write A LibraryCase class with main method that uses the previous classes to do the following:

* Create a Stock object
* Populate it with 3 Books objects and 3 Best Sellers
* Print the details of Best Sellers only
* Print for each book – its name and amount in stock
* Rent 1 regular book
* Rent 1 best seller
* Print current stock details again
* Return regular book
* Print current stock for the last time

**Adding Exceptions functionality:**

1. Create a NotInStockException class to be used as checked exception in Library system

* Add the following attribute: Book book
* Provide a constructor that takes Book instance and an error message

1. Update Stock.rentBook() method to throw NotInStockException in there are no copies left of the requested book (amount=0)
2. Update the client that uses this method accordingly – It should catch the error and print its details nicely to the screen