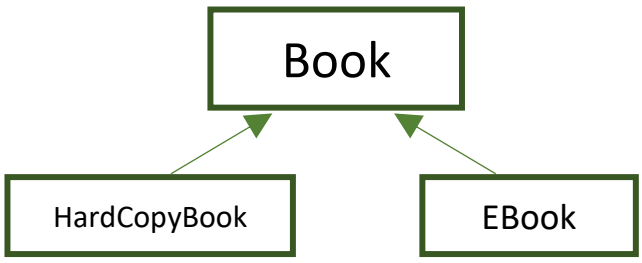
COMP 132: Advanced Programming
Self-Study Problems 2

Inheritance concepts

Imagine an online book store where you can buy hard copy books, or e-books.
The store also sells books from other suppliers.
Represent the books that can be bought in this store using the following class hierarchy:



Book class

Books sold from other suppliers are represented by the Book class.

➔ Book class should be in *store.book* package

➔ Each Book object stores the following information:

- *name:* A String
- *price:* A double
- *bookId:* an ISBN number; a String consisting of exactly 10 digits

➔ Each Book object should have the following methods:

- It has getter and setter methods for each field. The setter methods make sure that the fields have reasonable values, e.g., the *price* is nonnegative or the *bookId* is of the required format. Otherwise, the setters set the fields to some default value.
- A constructor with three arguments, corresponding to the three fields.
- A method *public void applyDiscount(double discountPercentage)* that reduces the book price by *discountPercentage* percent.
- A *public double getTotalCost()* method which returns the cost of the book, in this case it is the *price* value.
- A *toString()* method

HardCopyBook class

➔ *HardCopyBook* class should be in *store.book* package

➔ Each *HardCopyBook* is a special kind of *Book* object that has the following additional fields:

- *weight:* a double
- *shippingCost*: a double

➔ *HardCopyBook* objects should also have proper getters and setters, and they

should override Book's *toString* method so that the return value includes *weight* and *shippingCost* information as well as other information in the *Book* class. The *HardCopyBook* constructor should take five arguments corresponding to the five fields of this class and its superclass.

➔ *HardCopyBook* objects should also override Book's *getTotalCost() method.*

- public double *getTotalCost*() which returns the cost of buying (i.e., price) plus *shippingCost*.

EBook class

➔ *EBook* class should be in *store.book* package

➔ *EBook* objects represents e-books, and have the following additional fields:

- *fileSize*: a long, the number of bytes of the file representing this book
- *encodingFormat*: A String such as "pdf", "chm", "djvu", etc. representing what format the file is encoded in.
- *numDevicesAllowed*: an int, the maximum number of electronic devices that the customer can have copies of this e-book on.
- *numDevicesBeingUsed*: an int representing the number of devices that the customer already has copies of this e-book on. The default is 0.

➔ *EBook* objects should also have a proper constructor that takes in six arguments corresponding to the fields(excluding *numDevicesBeingUsed*). The *EBook* class should have getters and setters for all fields except the setter of *numDevicesBeingUsed,* since we don't want it to be modified unless a device is added or removed (i.e., in *addDevice*() and *removeDevice*()). Setters should do reasonable validity checking.

➔ *EBook* objects should have methods

- *public boolean addDevice()*
- *public boolean removeDevice()*

that, when they return "true", it means adding or removeing a device from the set of devices on which copies of the item reside was done successfully. When addition or removal fails, these methods return "false."

Write a class called ***ShoppingCart*** in the *store.shopping* package that represents a shopping cart with at most 10 entries.

- Use a *Book* array of 10 entries as one of ShoppingCart's fields. Call it *contents*. contents can store *Book*, *HardCopyBook*, or *EBook* objects.
- Use an integer field, *numEntriesInCart* (<=10) that represents the number of books already in the cart.
- Write getter and setter methods for all non-array fields **<u>if needed</u>** (note that *numEntriesInCart* should not be modified by user).
- Write a method *public boolean addBook(Book book)* that, if the cart has fewer than 10 entries, adds book to the shopping cart, and increases *numEntriesInCart* by 1. This method should return "true" if the addition is successful, "false" if the cart was full.

- Write a method *public boolean removeLastBook()* that removes the last book added to the cart and returns “true” if there is such an item. This method returns “false” if the cart is empty.
- Write a “*toString()*” method for the cart that uses the *toString()* method of the entries in the contents array. The returned String should also contain the total number, price and cost of the books in the shopping cart.

Write a Test class that creates a *ShoppingCart* object, fills it with 7 objects (a mix of *Book*, *HardCopyBook* and *EBook* objects) and then calls the *toString* method of the *ShoppingCart* object.