Nathan Cheung CSSE 374 2-15-15
CM 1586 Exam 3 Answers

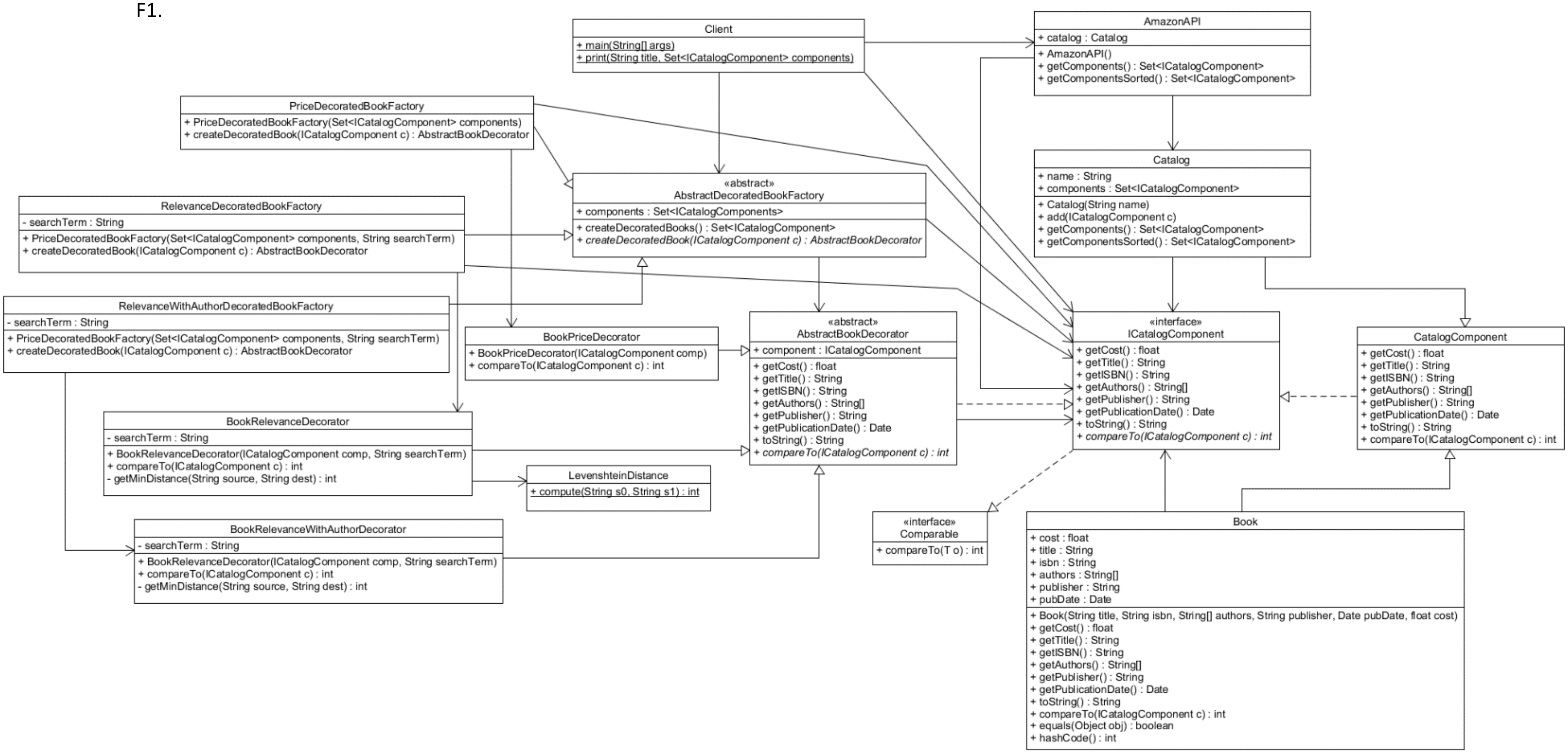
Q1.

The Sort objects implement the Template Method since the Comparator interface gives the algorithm but the Sort objects define how to compare the books.

Q2.

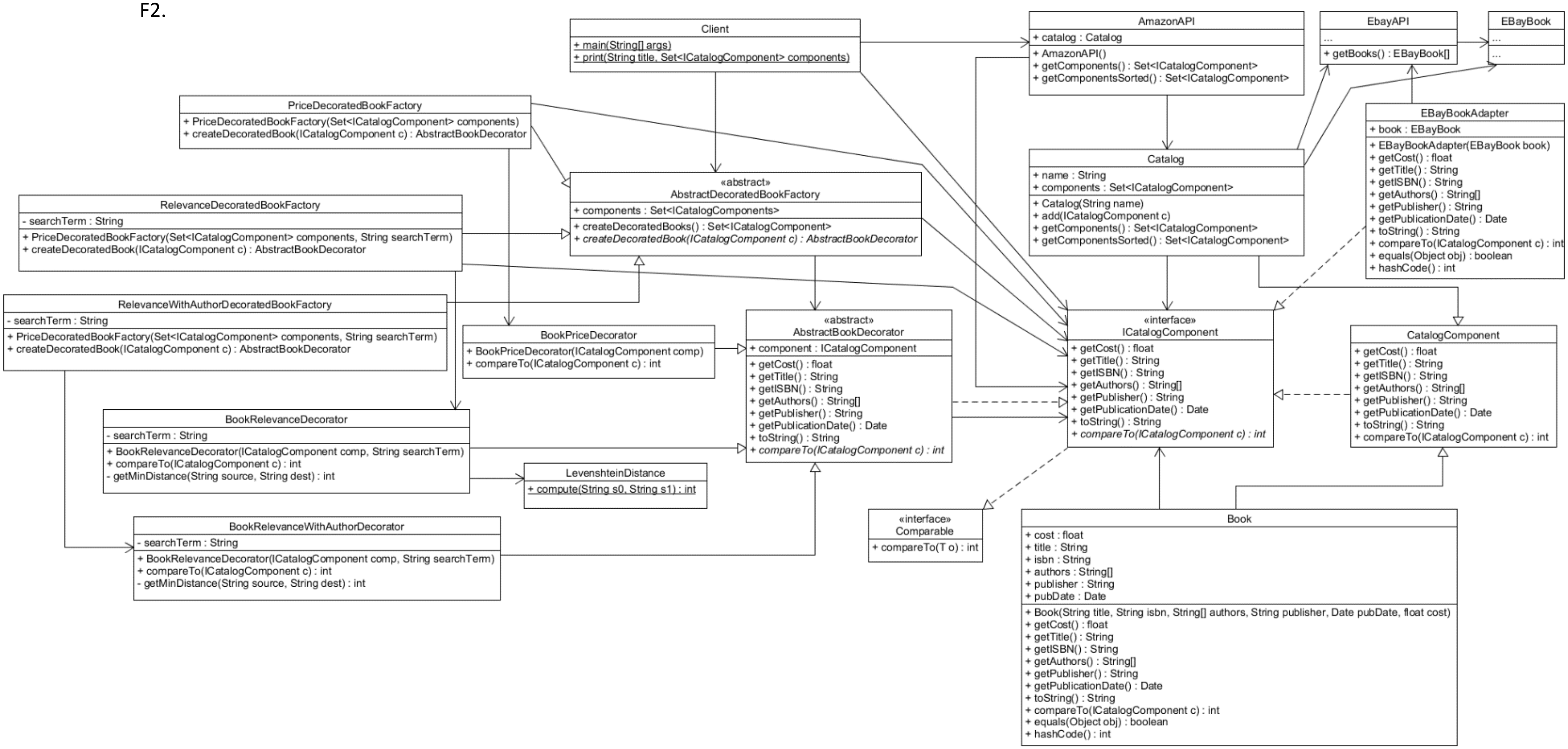
A new method has to be created for each sorting algorithm as well as the corresponding Sort object with the algorithm. This is pretty inefficient. If the client wants a new sorting algorithm, he must make a request to the Amazon team to implement it; he cannot implement the algorithm himself.

F1.


Client
+ main(String[] args)
+ print(String title, Set<ICatalogComponent> components)
AmazonAPI
+ catalog : Catalog
+ AmazonAPI()
+ getComponents() : Set<ICatalogComponent>
+ getComponentsSorted() : Set<ICatalogComponent>
PriceDecoratedBookFactory
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
«abstract»
AbstractDecoratedBookFactory
+ components : Set<ICatalogComponents>
+ createDecoratedBooks() : Set<ICatalogComponent>
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
Catalog
+ name : String
+ components : Set<ICatalogComponent>
+ Catalog(String name)
+ add(ICatalogComponent c)
+ getComponents() : Set<ICatalogComponent>
+ getComponentsSorted() : Set<ICatalogComponent>
RelevanceDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
RelevanceWithAuthorDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
BookPriceDecorator
+ BookPriceDecorator(ICatalogComponent comp)
+ compareTo(ICatalogComponent c) : int
«abstract»
AbstractBookDecorator
+ component : ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
«interface»
ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
CatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
BookRelevanceDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int
LevenshteinDistance
+ compute(String s0, String s1) : int
«interface»
Comparable
+ compareTo(T o) : int
BookRelevanceWithAuthorDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int
Book
+ cost : float
+ title : String
+ isbn : String
+ authors : String[]
+ publisher : String
+ pubDate : Date
+ Book(String title, String isbn, String[] authors, String publisher, Date pubDate, float cost)
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
+ equals(Object obj) : boolean
+ hashCode() : int

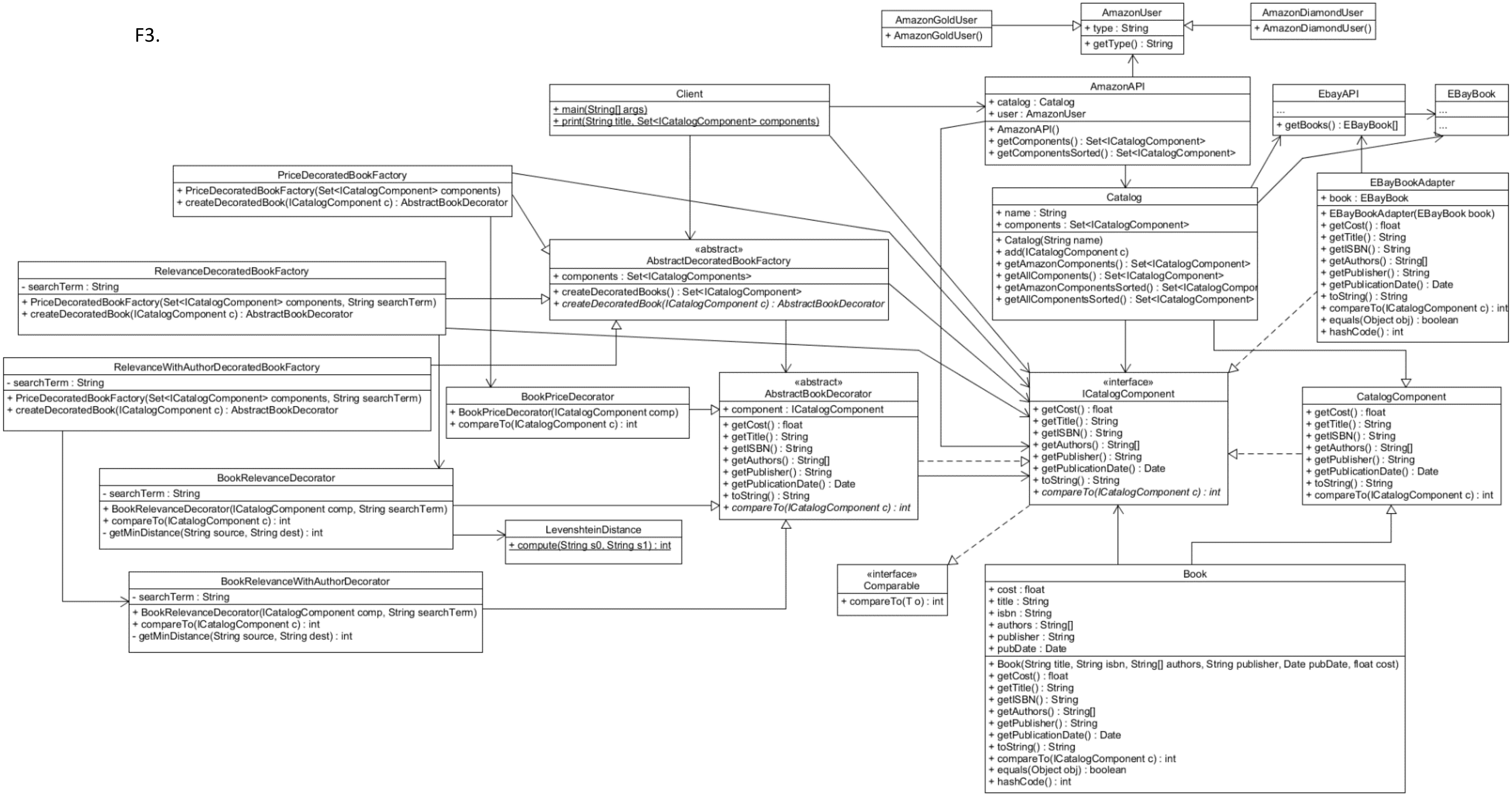
The current design implements the Composite, Abstract Factory, and Decorator patterns. The Composite pattern is applied to the ICatalogComponent, AmazonAPI, Book, and Catalog classes. The ICatalogComponent is the component interface. The Book class is the leaf defining all the behavior for the elements in the composition while the Catalog is the Composite. The AmazonAPI is the client. The Abstract Factory pattern applies to the AbstractDecoratedBookFactory. This abstract factory helps create AbstractBookDecorators. The concrete DecoratedBookFactories specify the concrete BookDecorators that they create. The Decorator pattern is applied to the AbstractBookDecorator as well as the concrete BookDecorators. The ICatalogComponent class acts as the abstract component and the decorators handle the implementation for compareTo which is how the Catalog sorts the Books.

F2.


Client
+ main(String[] args)
+ print(String title, Set<ICatalogComponent> components)
AmazonAPI
+ catalog : Catalog
+ AmazonAPI()
+ getComponents() : Set<ICatalogComponent>
+ getComponentsSorted() : Set<ICatalogComponent>
EbayAPI
...
+ getBooks() : EBayBook[]
EBayBook
...
...
PriceDecoratedBookFactory
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
EBayBookAdapter
+ book : EBayBook
+ EBayBookAdapter(EBayBook book)
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
+ equals(Object obj) : boolean
+ hashCode() : int
Catalog
+ name : String
+ components : Set<ICatalogComponent>
+ Catalog(String name)
+ add(ICatalogComponent c)
+ getComponents() : Set<ICatalogComponent>
+ getComponentsSorted() : Set<ICatalogComponent>
«abstract»
AbstractDecoratedBookFactory
+ components : Set<ICatalogComponents>
+ createDecoratedBooks() : Set<ICatalogComponent>
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
RelevanceDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
RelevanceWithAuthorDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
BookPriceDecorator
+ BookPriceDecorator(ICatalogComponent comp)
+ compareTo(ICatalogComponent c) : int
«abstract»
AbstractBookDecorator
+ component : ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
«interface»
ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
CatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
BookRelevanceDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int
LevenshteinDistance
+ compute(String s0, String s1) : int
«interface»
Comparable
+ compareTo(T o) : int
BookRelevanceWithAuthorDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int
Book
+ cost : float
+ title : String
+ isbn : String
+ authors : String[]
+ publisher : String
+ pubDate : Date
+ Book(String title, String isbn, String[] authors, String publisher, Date pubDate, float cost)
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
+ equals(Object obj) : boolean
+ hashCode() : int

The design now implements the Adapter pattern in addition to the other 3 patterns from F1. The Catalog gets the array of EBayBooks from the EbayAPI before using the EBayBookAdapter to make the EBayBooks all implement ICatalogComponent. Once the EBayBooks are adapted, it is added to a set containing the Amazon books, resulting in a set of all the books from both companies. This way the Client and Amazon's API don't have to handle the logic of making the EBayBooks compatible.

F3.


AmazonGoldUser
+ AmazonGoldUser()
AmazonUser
+ type : String
+ getType() : String
AmazonDiamondUser
+ AmazonDiamondUser()
Client
+ main(String[] args)
+ print(String title, Set<ICatalogComponent> components)
AmazonAPI
+ catalog : Catalog
+ user : AmazonUser
+ AmazonAPI()
+ getComponents() : Set<ICatalogComponent>
+ getComponentsSorted() : Set<ICatalogComponent>
EbayAPI
...
+ getBooks() : EBayBook[]
EBayBook
...
...
PriceDecoratedBookFactory
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
Catalog
+ name : String
+ components : Set<ICatalogComponent>
+ Catalog(String name)
+ add(ICatalogComponent c)
+ getAmazonComponents() : Set<ICatalogComponent>
+ getAllComponents() : Set<ICatalogComponent>
+ getAmazonComponentsSorted() : Set<ICatalogCompo
+ getAllComponentsSorted() : Set<ICatalogComponent>
EBayBookAdapter
+ book : EBayBook
+ EBayBookAdapter(EBayBook book)
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
+ equals(Object obj) : boolean
+ hashCode() : int
«abstract»
AbstractDecoratedBookFactory
+ components : Set<ICatalogComponents>
+ createDecoratedBooks() : Set<ICatalogComponent>
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
RelevanceDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
RelevanceWithAuthorDecoratedBookFactory
- searchTerm : String
+ PriceDecoratedBookFactory(Set<ICatalogComponent> components, String searchTerm)
+ createDecoratedBook(ICatalogComponent c) : AbstractBookDecorator
BookPriceDecorator
+ BookPriceDecorator(ICatalogComponent comp)
+ compareTo(ICatalogComponent c) : int
«abstract»
AbstractBookDecorator
+ component : ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
«interface»
ICatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
CatalogComponent
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
BookRelevanceDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int
LevenshteinDistance
+ compute(String s0, String s1) : int
«interface»
Comparable
+ compareTo(T o) : int
Book
+ cost : float
+ title : String
+ isbn : String
+ authors : String[]
+ publisher : String
+ pubDate : Date
+ Book(String title, String isbn, String[] authors, String publisher, Date pubDate, float cost)
+ getCost() : float
+ getTitle() : String
+ getISBN() : String
+ getAuthors() : String[]
+ getPublisher() : String
+ getPublicationDate() : Date
+ toString() : String
+ compareTo(ICatalogComponent c) : int
+ equals(Object obj) : boolean
+ hashCode() : int
BookRelevanceWithAuthorDecorator
- searchTerm : String
+ BookRelevanceDecorator(ICatalogComponent comp, String searchTerm)
+ compareTo(ICatalogComponent c) : int
- getMinDistance(String source, String dest) : int

The API now makes use of the Template method as well as a hook. The AmazonAPI's getComponents and getComponentsSorted methods now implement a hook that is dependent on the AmazonUser's type. If the user is an AmazonGoldUser, the catalog's getAmazonComponents and getAmazonComponentsSorted methods will be called by the API. If the user is an AmazonDiamondUser, the catalog's getAllComponents and getAllComponentsSorted methods will be called by the API.