COMP/EECE 4081

Final Exam

Fall 2012

Name: Answer Key

Rules:

- No potty breaks.
- Turn off cell phones.
- Closed book, closed note, closed neighbor.

Reminders:

- Verify that you have 18 pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

Additional Items:

- For questions that involve writing code, you may omit `import` statements unless specifically asked for them.

For each of the following multiple-choice questions, circle the (one) response that best answers the question.

1. [2pts] What is the typical relationship between coupling and cohesion?

   a. There is no relationship between coupling and cohesion.

   b. As cohesion increases, coupling increases.

   [✓ c.] As cohesion increases, coupling decreases.

2. [2pts] All else being equal, which is more desirable?

   a. Higher cohesion and higher coupling

   [✓ b.] Higher cohesion and lower coupling

   c. Lower cohesion and lower coupling

   d. Lower cohesion and higher coupling

   e. None of the above is more desirable than the others.

3. [2pts] All "real" software contains bugs. True or false?

   [✓ a.] True

   b. False

4. [2pts] Generally, in practice, developers exhaustively test software. True or false?

   a. True

   [✓ b.] False

The ethical theory of *utilitarianism* is based upon the principle of utility:

> **Principle of Utility:** An action is moral (or immoral) to the extent that it increases (or decreases) the total happiness of the affected parties. You may think of "happiness" as advantage, benefit, good, or pleasure, and "unhappiness" as disadvantage, cost, evil, or pain.

Now, consider the following ethically sticky situation. To the best of the BillEx corporation's ability, they strive to prevent billing errors; however, such errors sometimes occur. In such cases, customers must spend time and effort to straighten out the mistake.

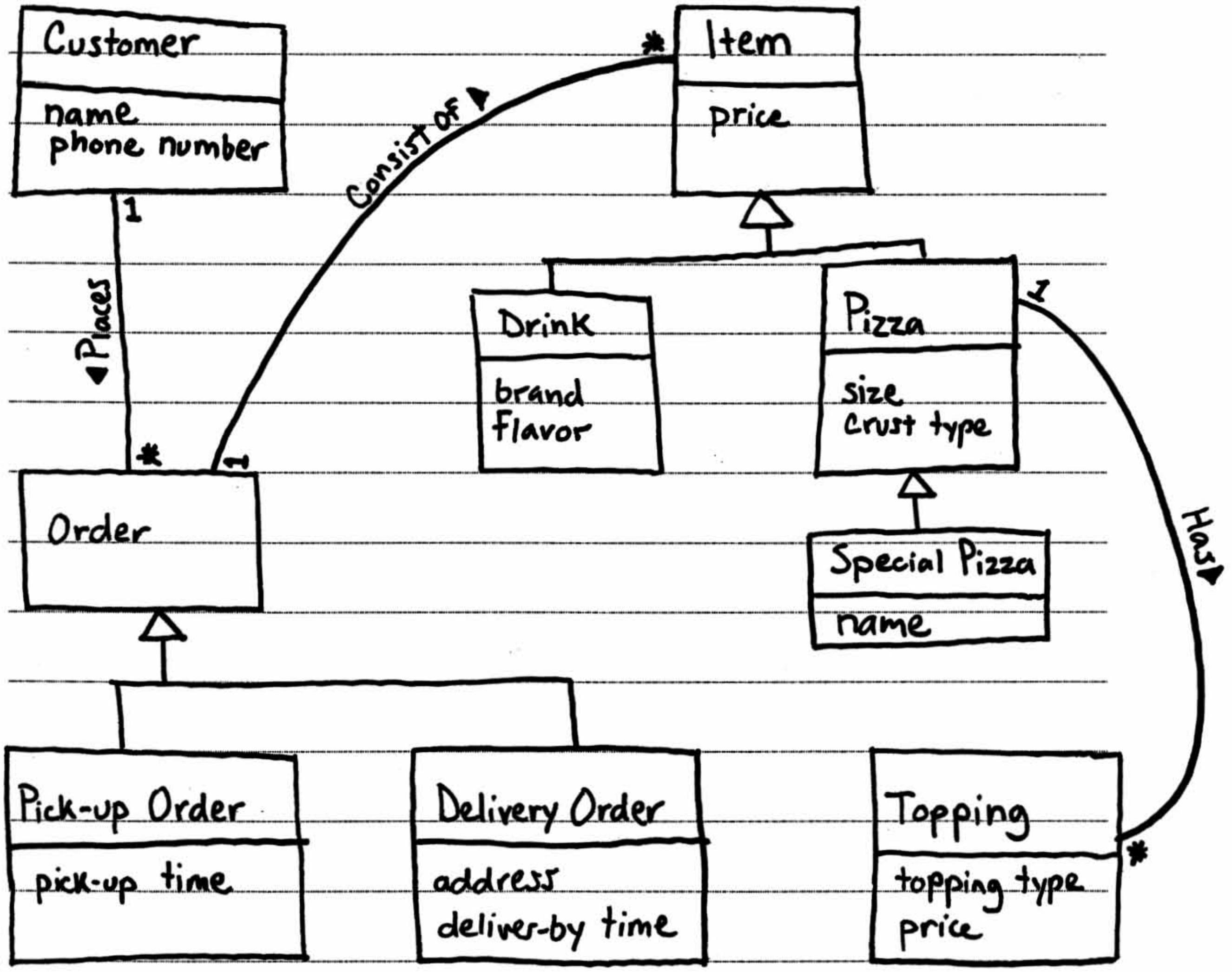
5. [12pts] From the standpoint of utilitarianism, should BillEx compensate customers inconvenienced by billing errors? Thoroughly justify your answer. (Your grade on this question will be based largely on the quality of your justification.)

[Many possible valid responses.]

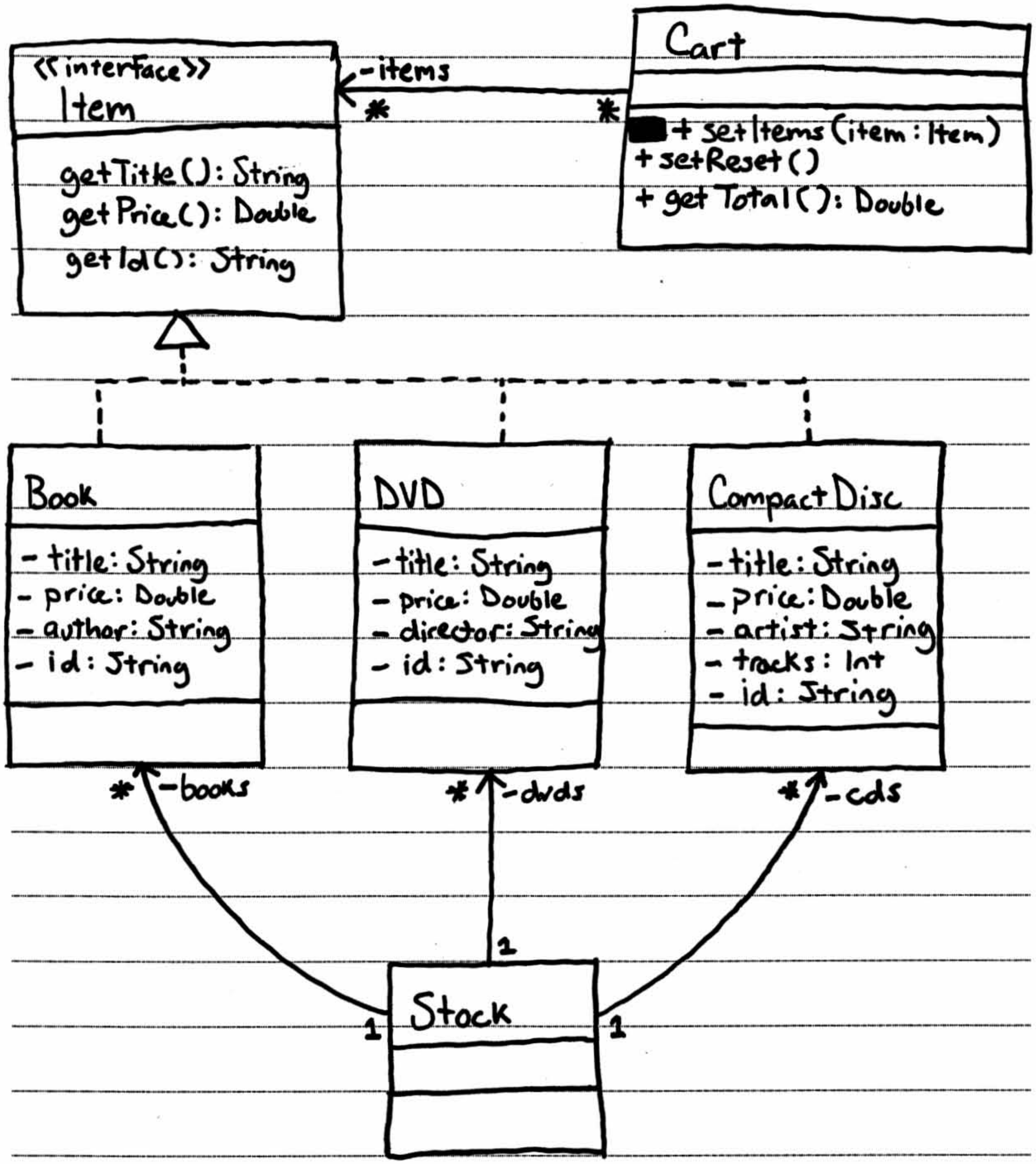
(There is more room on the next page.)

6. [10pts] Imagine that you are tasked with developing a system for a pizza shop. Given the following description, create a Domain Model (in the form of a UML class diagram). Include all conceptual classes, attributes, associations, and generalization relationships mentioned in the descriptions. Label all associations and include all multiplicities.

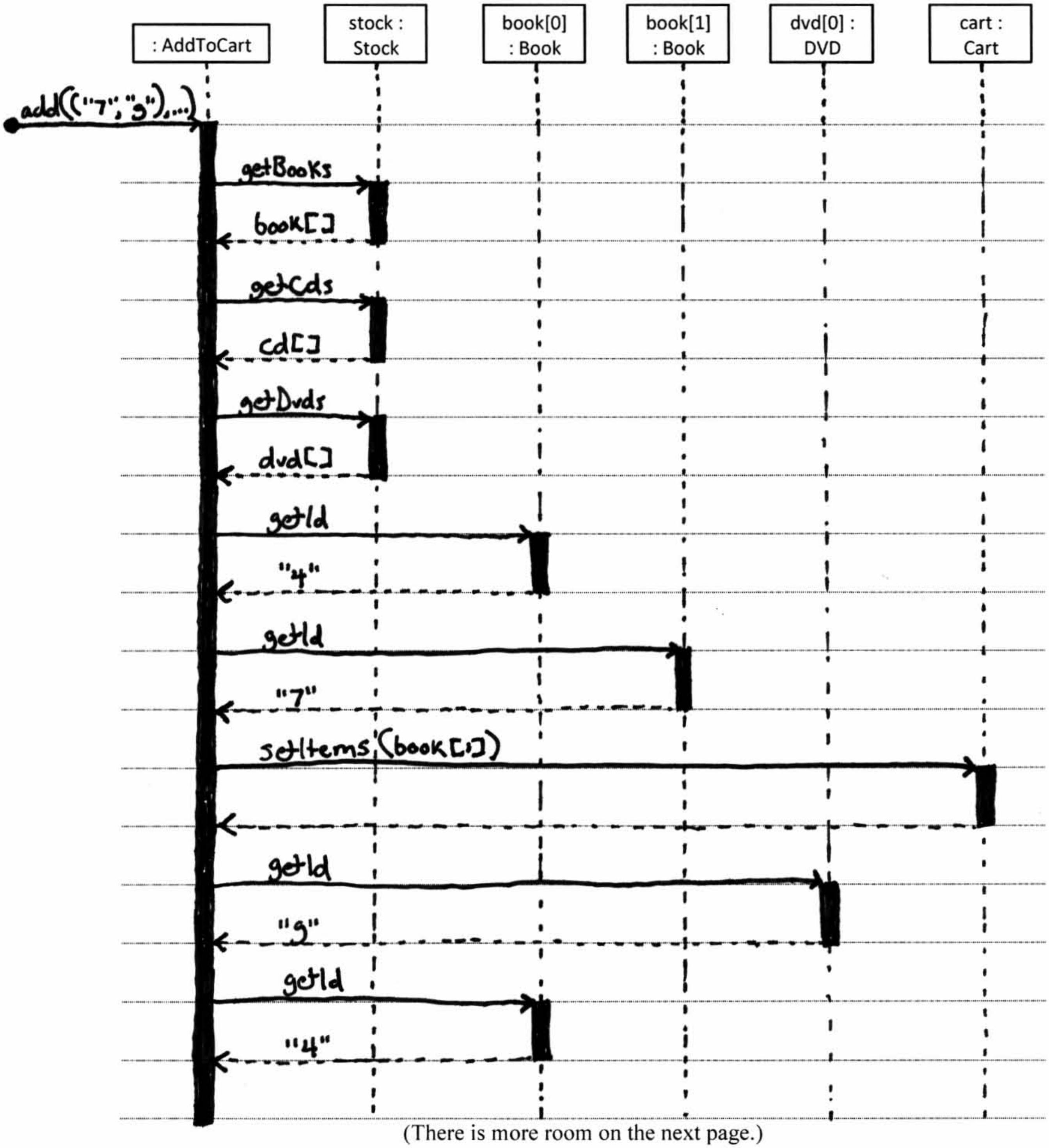
A customer places orders. A customer has a name and phone number. There are two types of orders: pick-up and delivery. A pick-up order has a pick-up time. A delivery order has an address and deliver-by time. All orders consist of a set of items. There are two types of items: pizzas and drinks. All items have a price. A pizza has a size and a crust type. A pizza also has a number of toppings. A topping has a topping type and a price. Some pizzas are special pizzas that have a name (e.g., "Hawaiian" or "Meat Lovers"). A drink has a brand and a flavor.



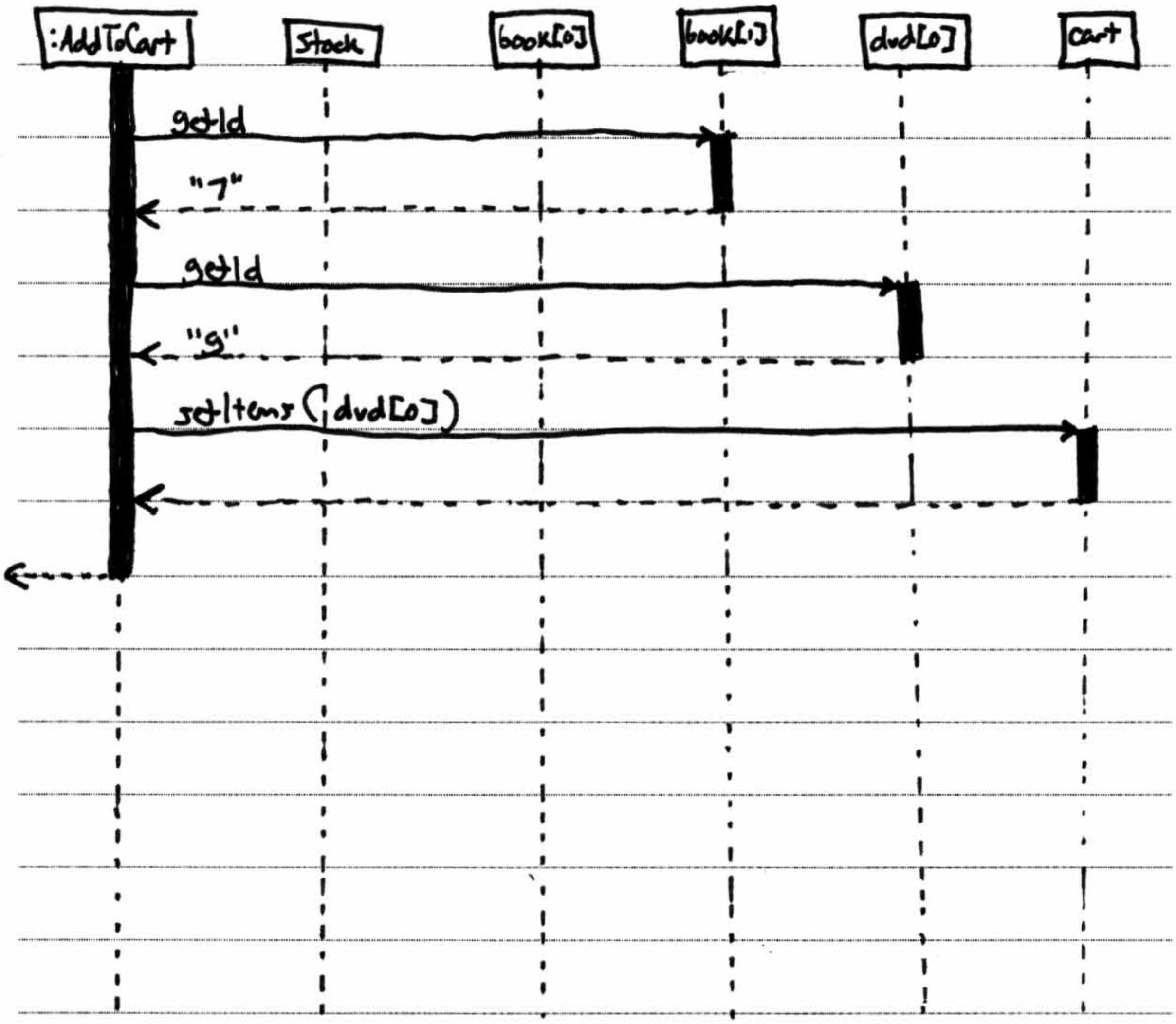
7. [13pts] Reverse engineer a design class diagram for the Media"R"Us.com system (page 16). Include only the following classes/interfaces in your diagram: Item, Book, DVD, CompactDisc, Cart, and Stock. Model completely all attributes, operations, associations, visibilities, multiplicities, generalizations, etc. You may omit constructors and simple getter/setter operations that do nothing more than return/assign an instance variable.



8. [12pts] Consider the object diagram (snapshot) of a running Media"R"Us system (page 19). Complete the following sequence diagram that models the interactions that would result from an execution of AddToCart's add() method when method parameter ids = ("7", "g"). Model only interactions among the objects given below—be careful not to miss any! Explicitly model returns and return values, and execution specifications (aka activation bars).



(There is more room on the next page.)


:AddToCart
Stock
book[0]
book[1]
dvd[0]
cart
getId
"7"
getId
"9"
setItems(dvd[0])

9. [5pts] Consider the total cost of customer's purchase of books, DVDs, and/or CDs. Which Media"R"Us class is an information expert with respect to the total cost? Why was that class chosen to be the information expert?

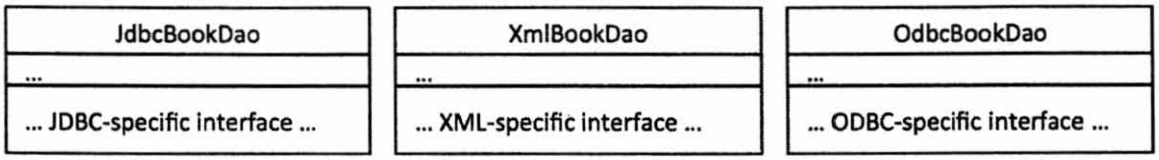
Cart is the information expert for total cost of purchase.

Cart was chosen to be information expert because it is the class with all the information necessary to fulfill that responsibility. That is, a Cart knows all the Items in a purchase, and each item knows its unit cost.

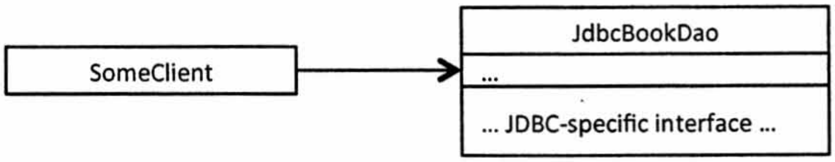
10. [2pts] To which component of an MVC architecture should the Item, Cart, and Stock classes belong? Give the full name for the component (not just the letter).

Model

There are many different ways that you might want to store Media"R"Us items in persistent storage. For example, you might store items to an XML file, or store them in a DB (via the JDBC interface), or transmit them via TCP to some other type of server. Imagine that the Book class has a corresponding data access object (DAO) class for each storage technology, and each DAO class has a slightly different interface. For example:


JdbcBookDao
...
... JDBC-specific interface ...
XmlBookDao
...
... XML-specific interface ...
OdbcBookDao
...
... ODBC-specific interface ...

Now, imagine some client code implements a media sales system and uses the JdbcBookDao class to create/read/update/delete stored Book objects:

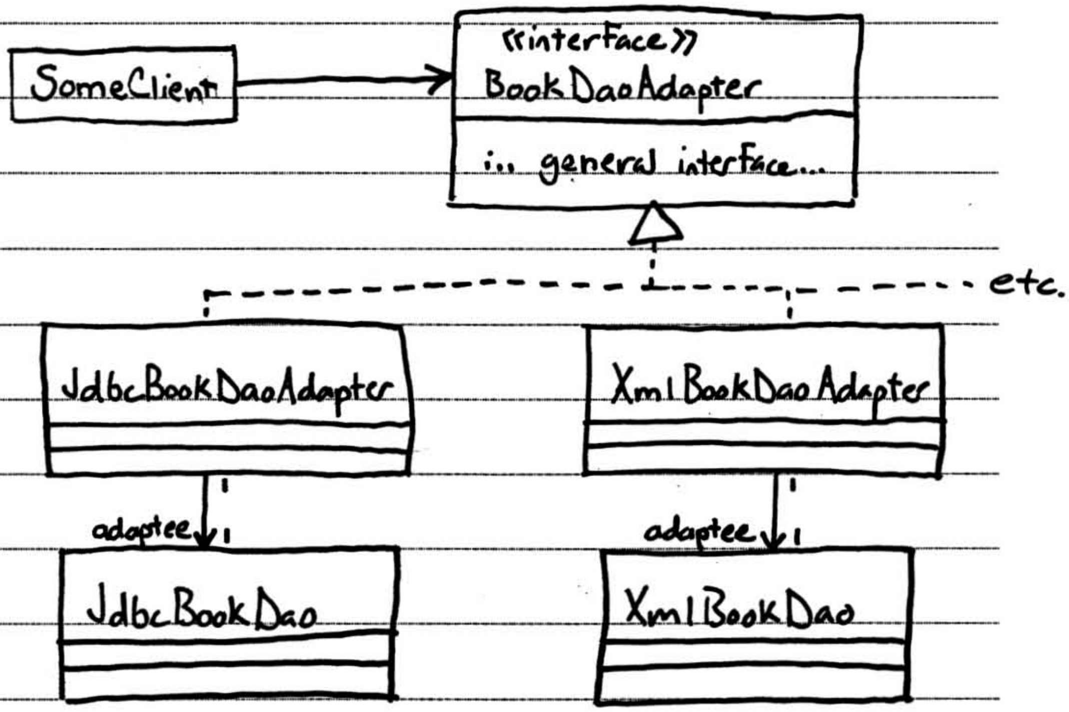


11. [5pts] What drawback does the SomeClient/JdbcBookDao design have if the choice of storage technology tends to be an unstable design decision (i.e., prone to change)?

The drawback is that you need to modify the SomeClient code wherever you change storage technology.

12. [10pts] How might you improve the design to overcome this drawback? Explain your design with both words and a design class diagram. Name two of the design patterns discussed in class, and explain how each contributes to your design.

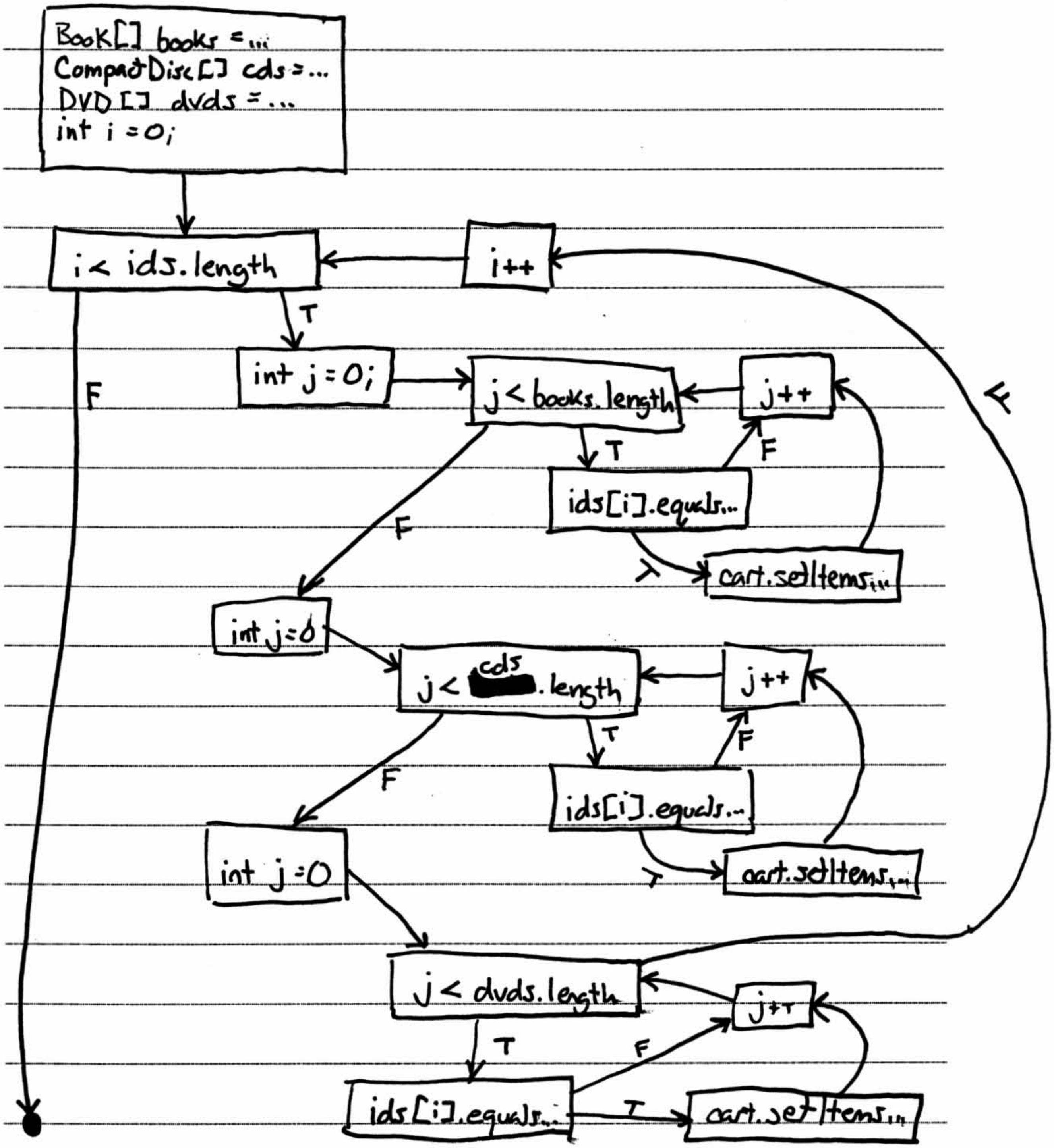
To improve the design, I would decouple the SomeClient and JdbcBookDao classes by inserting an adapter class, like this:



This design applies 3 main patterns:

- Indirection: It inserts a level of indirection between client and BookDao.
- Protected Variation: Creates stable adapter interface around unstable BookDao interfaces.
- Polymorphism: BookDaoAdapter interface operations are polymorphic, so different storage technologies can be swapped in and out.

13. [10pts] Draw a control flow graph (CFG) for AddToCart's add() method. (See the example of a for-loop CFG on page 20.) You may shorten long lines of code using ellipses (i.e., "…"s); for example: `Book[] books = …`



In the next two questions, you will use your CFG to define test suites. Because the id is the only part of an Item we care about in these problems, you may use the following shorthand notation:

- ids = ("id1", "id2", "id3")
  - Specifies an array of 3 id strings.
- cart = ("id4", "id5")
  - Specifies a cart with two items with id="id4" and id="id5", respectively.
- stock = (Books → ("id6", "id7"), CDs → ("id8"), DVDs → ("id9", "id10"))
  - Specifies a stock with 2 books, 1 CD, and 2 DVDs.

To simplify matters, you may omit the expected output from your text cases.

14. [5pts] Specify a test suite that achieves statement coverage of AddToCart's add() method.

Test case #1:

ids = ("a", "b", "c")

cart = ()

stock = (Books → ("a"), CDs → ("b"), DVDs → ("c"))

15. [5pts] Specify a test suite that achieves condition coverage of the AddToCart's add() method.

The suite From question 15 will work here.

16. [3pts] How may test cases would a suite have that does path coverage (with max of 1 iteration per loop) of add()?

a. 1

b. 2

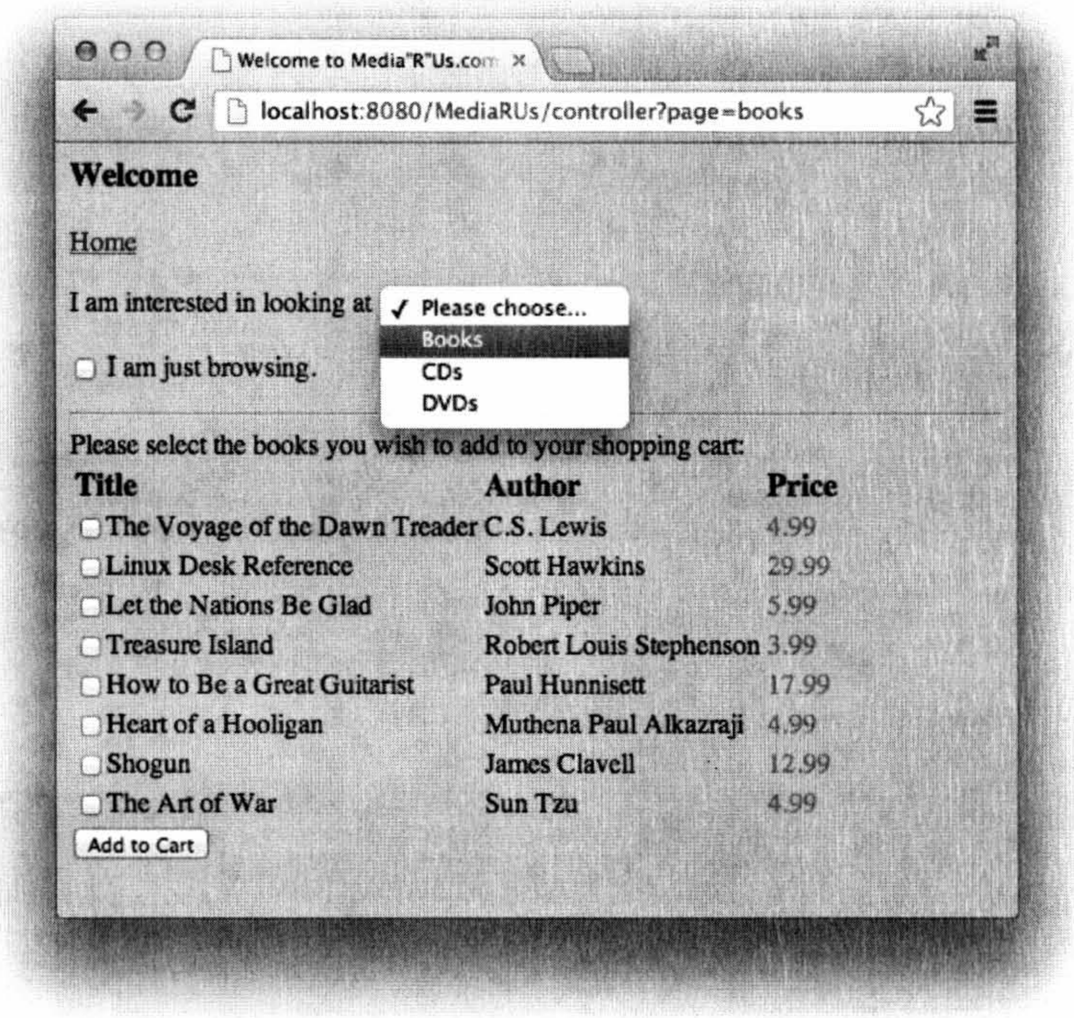
c. 4

d. 7

[✓ e.] More than 7

Media“R”Us.com Source Code

The following source code belongs to the code base for a media sales system. You might use this code to build an online media store like this:


Welcome to Media"R"Us.com
localhost:8080/MediaRUs/controller?page=books
✓ Please choose...
Books
CDs
DVDs
Add to Cart

Welcome

Home

I am interested in looking at

□ I am just browsing.

Please select the books you wish to add to your shopping cart:

Title	Author	Price
□ The Voyage of the Dawn Treader	C.S. Lewis	4.99
□ Linux Desk Reference	Scott Hawkins	29.99
□ Let the Nations Be Glad	John Piper	5.99
□ Treasure Island	Robert Louis Stephenson	3.99
□ How to Be a Great Guitarist	Paul Hunnisett	17.99
□ Heart of a Hooligan	Muthena Paul Alkazraji	4.99
□ Shogun	James Clavell	12.99
□ The Art of War	Sun Tzu	4.99

Media items include books, CDs, and DVDs. The code includes functionality for a cart to which shoppers can add their candidate purchases, and functionality for keeping track of the store’s stock of media items.

```
public interface Item {
       String getTitle();
       double getPrice();
       String getId();
}
```

```
public class Book implements Item{
       //Instance variables
       private String title;
       private double price;
       private String author;
       private String id;

       //Methods
       public String getId(){ return id; }
       public String getTitle(){ return title; }
       public double getPrice(){ return price; }
       public String getAuthor(){ return author; }

       //constructors
       public Book(){}

       public Book(String t, double p,String a, String i){
              title=t;
              price=p;
              author=a;
              id=i;
       }
}
```

```
public class DVD implements Item{
       //Instance variables
       private String title;
       private double price;
       private String director;
       private String id;

       //Methods
       public String getId(){ return id; }
       public String getTitle(){ return title; }
       public double getPrice(){ return price; }
       public String getDirector(){ return director; }

       //constructors
       public DVD(){}

       public DVD(String t, double p,String d, String i){
              title=t;
              price=p;
              director=d;
              id = i;
       }
}
```

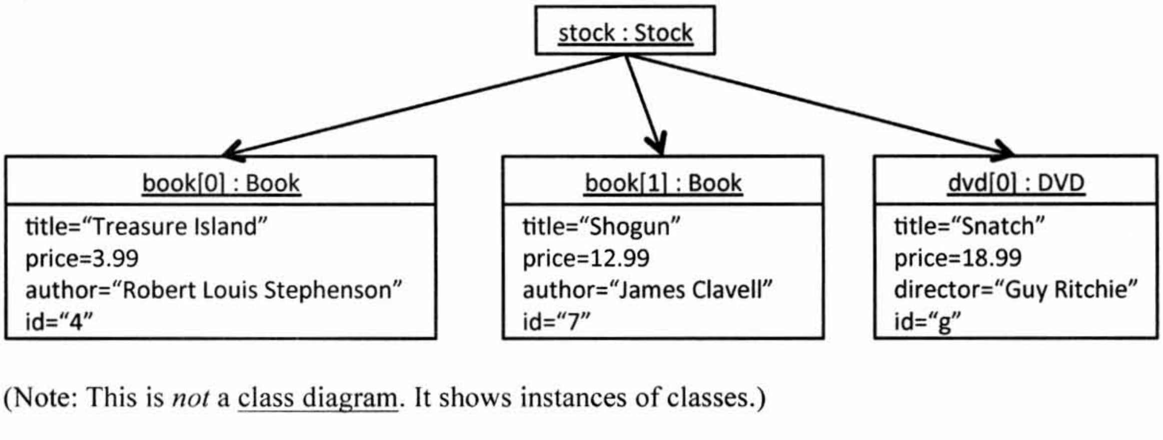
```
public class CompactDisc implements Item{
        //Instance variables
        private String title;
        private double price;
        private String artist;
        private int tracks;
        private String id;

        //Methods
        public String getId(){ return id; }
        public String getTitle(){ return title; }
        public double getPrice(){ return price; }
        public String getArtist(){ return artist; }
        public int getTracks(){ return tracks; }

        //constructors
        public CompactDisc(){}

        public CompactDisc(String t, double p, String a,
                           int tr, String i){
                title=t;
                price=p;
                artist=a;
                tracks=tr;
                id = i;
        }
}
```

```
public class Cart {
        private Item[] items = new Item[0];

        public Item[] getItems() { return items; }

        public void setItems(Item item) {
                Vector v = new Vector(1);
                for (int i=0;i<items.length;i++){
                        v.add(items[i]);
                }
                v.add(item);
                items = (Item[])v.toArray(items);
        }

        public void setReset() { items = new Item[0]; }

        public double getTotal() {
                double total = 0;
                for (int i=0;i<items.length;i++){
                        total+=items[i].getPrice();
                }
                return total;
        }
}
```

```
public class Stock{
        private Book[] books;
        private CompactDisc[] cds;
        private DVD[] dvds;

        public Stock() { ... }

        public Book[] getBooks() { return books; }
        public CompactDisc[] getCds() { return cds; }
        public DVD[] getDvds() { return dvds; }
}
```

```
public class AddToCart {

        public void add(String[] ids, Cart cart, Stock stock) {
                Book[] books = stock.getBooks();
                CompactDisc[] cds = stock.getCds();
                DVD[] dvds = stock.getDvds();

                for (int i = 0; i < ids.length; i++) {
                        for (int j = 0; j < books.length; j++) {
                                if (ids[i].equals(books[j].getId())) {
                                        cart.setItems(books[j]);
                                }
                        }
                        for (int j = 0; j < cds.length; j++) {
                                if (ids[i].equals(cds[j].getId())) {
                                        cart.setItems(cds[j]);
                                }
                        }
                        for (int j = 0; j < dvds.length; j++) {
                                if (ids[i].equals(dvds[j].getId())) {
                                        cart.setItems(dvds[j]);
                                }
                        }
                }
        }
}
```

Media“R”Us Object Diagram



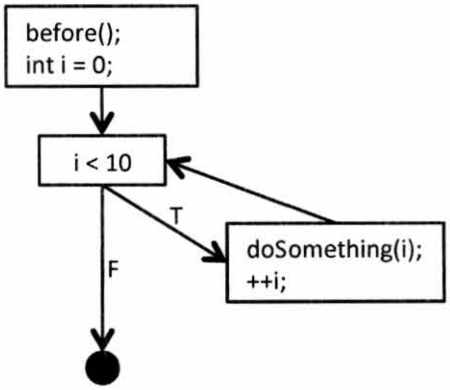
(Note: This is *not* a class diagram. It shows instances of classes.)

Control Flow Example

For Loop

```
before();
for (int i = 0; i < 10; ++i) {
  doSomething(i);
}
```

Control Flow Graph


before();
int i = 0;
i < 10
T
doSomething(i);
++i;
F