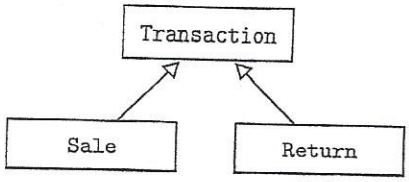
. Consider a program that keeps track of transactions in a large department store. Both sales and returns are recorded. Three classes—`Transaction`, `Sale`, and `Return`—are used in the program, related as in the following inheritance hierarchy:



The diagram shows that both the `Sale` and `Return` classes are subclasses of the `Transaction` class.
The `Transaction` class is defined below:

```
public class Transaction
{
    private String myDescription;
    private int myNumItems;
    private double myItemCost;
    public static final double TAX_RATE = 0.07;

    //constructor
    public Transaction(String description, int numItems,
                       double itemCost)
    {
        myDescription = description;
        myNumItems = numItems;
        myItemCost = itemCost;
    }

    //accessors
    public String getDescription()
    { return myDescription; }

    public int getNumItems()
    { return myNumItems; }

    public double getItemCost()
    { return myItemCost; }

    public double getTotal()
    {
        double cost = myNumItems * myItemCost;
        double tax = cost * TAX_RATE;
        return cost + tax;
    }
}
```

(a) Write the code for the `Sale` class. Each `Sale` includes

- A description of the item being sold.
- The number of this item being sold.
- The cost of this item.
- Whether the sale is cash or credit, stored as a boolean variable.
- A 10 percent discount for cash, with 10 percent stored as a final variable.

When a new `Sale` is created, it must be assigned an item description, the number being sold, the cost of this item, and whether the sale is cash or credit. Operations on a `Sale` include the following:

- Retrieve the description of the item being sold.
- Retrieve the quantity of the item being sold.
- Retrieve the cost of the item being sold.
- Retrieve whether the sale is cash or credit.
- Calculate the total for the sale. In calculating this total, a 10 percent discount for paying cash should be applied to the cost before the tax is calculated. Hint: Discount is discount rate × cost.

Write the code for the `Sale` class below.

(b) A class called `DailyTransactions` has the following private instance variable:

```
/** The list of all transactions in a single day, including
 *  sales and returns */
private Transaction[] allTransactions;
```

Write `findTransactionAverage`, a method for the `DailyTransactions` class, which computes the average of all transactions in a given day. The transactions are contained in the array `allTransactions`, where each object is a `Sale` or `Return`. The method `findTransactionAverage` should

- Compute the total for all transactions.
- Divide by the number of transactions. (You may assume that there's at least one transaction.)
- Return the average.

Note that when an item is returned to the store, the amount paid is returned to the customer. For this reason, the `getTotal` method in the `Return` class returns a *negative* quantity.

Complete `findTransactionAverage` below:

```
/** @return the transaction average for one day
 *  Precondition:  allTransactions contains the day's transactions,
 *                 each of which may be a Sale or a Return.
 */
public double findTransactionAverage()
```