1.00/1.001

Introduction to Computers and Engineering Problem Solving

Fall 2010 - Final Exam

Name: **MIT Email:** **TA:** **Section:**	

You have 3 hours to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary packages have already been imported. You may only add code inside the boxes. The code written outside the boxes may not be altered in any way. Good luck!

Question 1	**/ 10**
Question 2	**/ 10**
Question 3	**/ 30**
Question 4	**/ 20**
Question 5	**/ 15**
Question 6	**/ 15**
Total	**/ 100**

Question 1 – Sorting (10 points)

Below is part of the code for class book. Implement the Comparable interface for this class. The sort order should be first alphabetically by author, then by number of pages (largest number first), and then by book height (largest height first). Use the generic version of the Comparable interface.

```
public class Book implements Comparable<Book> {

      private double height;
      private int pages;
      private String author;
      private String title;

      public Book(String title, String author, double height,
                                                            int pages){
            this.title=title;
            this.author=author;
            this.height=height;
            this.pages=pages;
      }

      public double getHeight() {return height;}
      public int getPages() {return pages;}
      public String getAuthor() {return author;}
      public String getTitle() {return title;}

      public int compareTo(Book b){

      }
}
```

Question 2 – Recursion (10 points)

A less known contemporary of Fibonacci, Notsofibbi, invented a new sequence to make himself famous. In this sequence, each term is the product of the preceding two terms. The first terms of the Notsofibbi sequence are therefore: 2, 3, 6, 18, 108, 1944…

Complete the notso(int n) method, which returns the nth Notsofibbi number using a recursive algorithm. The sequence starts at index 0, which means that:

`notso(0)` returns 2

`notso(1)` returns 3

...

You may assume that the method is always called with a non-negative integer argument.

```
public static int notso(int n){

}
```

Question 3 – Class design (30 points)

We want to model a store. Someone has already written the following abstract `Item` class:

```
public abstract class Item {
     protected int id;
     protected double price; // what a customer pays a store
     protected double cost;  // cost of producing an item

     public Item(int i, double p, double c){
          id = i;
          price = p;
          cost = c;
     }

     public double getProfit() {return price - cost;}

     public int getId() {return id;}

     public abstract void checkItem();
}
```

3.a Many items in a store are perishable. Write an interface called `Perishable` that defines a "near expiration" threshold `NEAREXP` to be 3 days, and a boolean method called `nearExpired()` that determines if something should be marked as being near expiration.

3.b Next, we want to model food items, which are perishable. Write a class `Food` that inherits from the code above. A `Food` object should keep track of its ID, price, cost, shelf life, and the number of days it has already spent on the store's shelf. A food item's shelf life is always 10 days in this problem. All other values are assigned from the constructor's parameters. Allow future classes to inherit data members in Food.

Implement all necessary methods in the `Food` class. If a food item is within `NEAREXP` days of its shelf life, it is marked as near expired. When a store manager "checks" a food item, a food item that is marked near expired will be sold at a 20% discount.

```
public class Food                                  {
```

3.c Now create a new class called `Fruit`. A fruit is a type of food, so inherit appropriately. When checking a fruit, if the fruit is near expired, its price will be reduced by half.

```
public class Fruit                                              {

}
```

3.d The store also sells electronics. All electronics in this store come with batteries. When checking an electronic item, if the battery capacity of the item is below 1500 mAh, then the price of the item is discounted by $2. Fill in the code below with the necessary methods. The battery capacity data member should be assigned from a parameter given to the constructor.

```
public class Electronics extends Item {
     private double battCap;

}
```

3.e We will now implement a store. This store currently sells only three items: an apple (with ID 4001, price of $0.60, cost of $0.10, and having been on the shelf for 6 days), a box of cereal (with ID 4002, price of $4.00, cost of $0.20, and having been on the shelf for 8 days), and a flashlight (with ID 9001, price of $10.00, cost of $5.00, and capacity of 2500 mAh). Add all items to the inventory. Then write the code that goes through the inventory, checks every item, computes the profit subtotal, and if it is perishable, prints out its ID and whether it is near expiration or not. Finally, print out the total store profit if everything sells today.

```
public class StoreTest {
  public static void main(String[] args) {
    ArrayList<Item> inventory = new ArrayList<Item>();
    inventory.add(new                                  );
    inventory.add(new                                  );
    inventory.add(new                                  );



  }
}
```

Question 4 - Dot Product (20 points)

In this question, you will implement a method to find the transpose of a matrix and a method to compute the dot product of two matrices. Recall the matrix class from lecture:

```
public class Matrix {

     private double[][] data;

     public Matrix(int m, int n) {data = new double[m][n];}

     public int getNumRows() {return data.length;}

     public int getNumCols() {return data[0].length;}

     public double getElement(int i, int j) {return data[i][j];}

     public void setElement(int i, int j, double val) {
          data[i][j] = val;
     }

     public Matrix mult(Matrix b) {
          Matrix result = null;
          int nrows = data.length;
          int ncols = data[0].length;
          if(ncols == b.data.length) {
               result = new Matrix(nrows, b.data[0].length);
               for(int i=0; i<nrows; i++)
                    for(int j=0; j<result.data[0].length; j++) {
                         double t = 0.0;
                         for(int k=0; k < ncols; k++)
                              t += data[i][k] * b.data[k][j];
                         result.data[i][j]= t;
                    }
          }
          return result;
     }

     public Matrix getTranspose(){
          //Code completed in Part a.
     }

     public double getDotProduct(Matrix m){
          //Code completed in Part b.
     }
}
```

4.a

First, we will write a method to find the transpose of a matrix.

A matrix transpose can be found by writing the rows of a matrix as the columns of its transpose. For example:

$$\text{If } A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \text{ then its transpose } A^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}.$$

Complete the method getTranspose() that finds the transpose of a Matrix. **Any matrix has a transpose (not just square matrices).**

```
public Matrix getTranspose()
{




}
```

4.b

Now we will write a method for finding the dot product of two vectors (or row matrices).

One method for calculating the dot product of two matrices is the following:

$$\mathrm{A} \cdot \mathrm{B} = \mathrm{A}^{\mathrm{T}}\mathrm{B}$$

where A^T is the transpose of the matrix A.

Using the getTranspose() method from Part a, complete the getDotProduct(..) method below for calculating the dot product of two matrices. Remember: You can only find the dot product of two vectors (or row matrices) of the same size. If the matrix argument (`Matrix m`) is not the same size as the matrix this method is being called on, this method should throw an *ArithmeticException* that will print to the console "Dot Product Requires Vectors of the Same Size!"

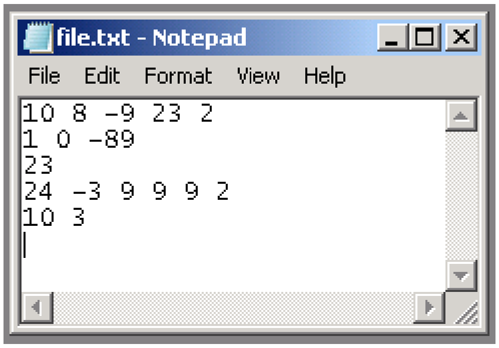
```
public double getDotProduct(Matrix m)
{




}
```

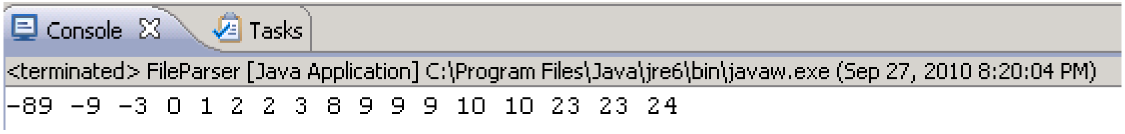
Question 5 – Streams (15 points)

You are given a text file which consists of a number of lines. Each line contains a varying number of integers, separated by a space. An example file looks like this:



```
10 8 -9 23 2
1 0 -89
23
24 -3 9 9 9 2
10 3
```

Write a method to read in all integers stored in a file and print them to the console in ascending order. With the above input file, your program should print:



```
<terminated> FileParser [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Sep 27, 2010 8:20:04 PM)
-89 -9 -3 0 1 2 2 3 8 9 9 9 10 10 23 23 24
```

You can assume that there are no format errors in the input file and the total number of integers stored in the file is less than 1000. You may assume appropriate packages are already imported and you may write helper methods if you feel necessary. Your program should handle any input/output exception that may arise.

```
}

public static void readAndPrint(String fileName){

}
```

Question 6 - Linked Lists (15 points)

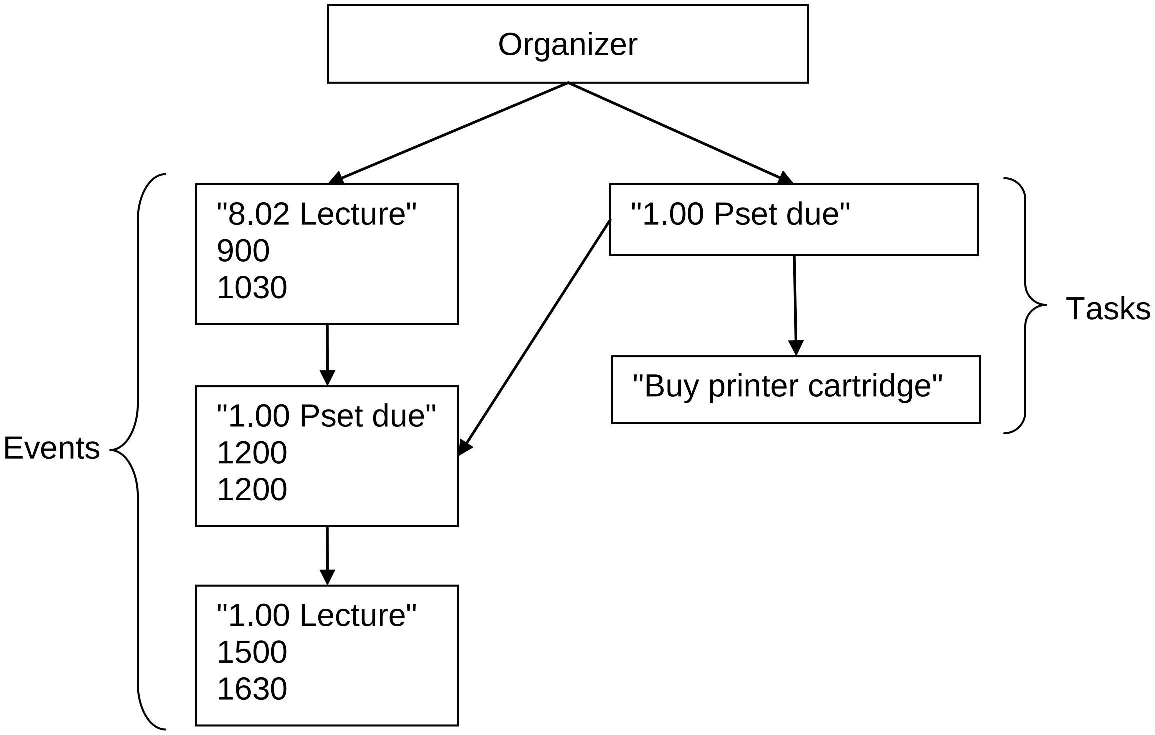
You are developing an electronic personal organizer that uses linked lists to handle tasks and events

The organizer keeps two lists: a list of events and a list of tasks.

Each event has a name, a start time and an end time. The start and end times are represented as integers (e.g. 1030 means 10:30am, 1500 means 3:00pm). The list of events is maintained in chronological order, with the earliest event first in the list. In this problem, we assume that there is no time conflict between events: two events may be scheduled back-to-back, but they never overlap. The list only contains events for one particular day.

Each task has a name, and a task may have a deadline. When a task has a deadline, it is convenient for the user to see the deadline in the schedule of events. To do this, an event representing the deadline is added to the event list. Such an event has the same name as the task and has start and end times both equal to the task deadline. The task list is in no particular order. Assume that task deadlines do not fall within oter events.

In the example below, an MIT student has an 8.02 lecture from 9am to 10:30am and a 1.00 lecture from 3pm to 4:30pm. He/she has a 1.00 homework due at 12noon. He/she also needs to buy a printer cartridge, but there is no rush for that.



Part of the `Organizer` class has already been written. It has two inner classes to represent events and tasks. You need to implement the `addTaskWithDeadline()` method, which will be used to add a task and its corresponding deadline to the organizer lists. You will write the method in 3 steps:

6.a Create the `Task` and `Event` objects that need to be added to the organizer lists. Make sure to create the appropriate relationship between those objects.

6.b Add the `Task` object to the task list.

6.c Add the `Event` object to the event list, keeping the list in chronological order.

```
public class Organizer {

     private Event firstEvent;
     private Task firstTask;

     private class Event{

          private String name;
          private int start;
          private int end;
          private Event next;

          public Event(String n, int s, int e){
               name = n;
               start = s;
               end = e;
               next = null;
          }
     }

     private class Task{

          private String name;
          private Event deadline;
          private Task next;

          public Task(String n){
               name = n;
               deadline = null;
               next = null;
          }
     }

// Organizer class continues next page
// takes in the name of the task (n) and the deadline (d)
public void addTaskWithDeadline(String n, int d){

     // a) create Task and Event objects





     // b) add Task object to task list





     // c) add Event object to event list





     }
}
```