1110 - Lab 10
Due: 11:59PM -- 10/31/12

Part 1

The `DateFormat` class of the standard Java library has a method

```
Date parse(String source)
```

that converts a string such as

```
"Nov 4, 2009 8:14 PM"
```

into an object of type `Date`:

```
DateFormat formatter = DateFormat.getDateTimeInstance(
      DateFormat.MEDIUM, DateFormat.SHORT);
String source = . . .;
Date d = formatter.parse(source);
```

If the source string is not in the correct format, then the `parse` method throws a `ParseException`, which is a checked exception. Be sure to review the format for dates that is described in the Java API for class `DateFormat`. Pay particular attention to the constants `DateFormat.SHORT` and `DateFormat.MEDIUM` because we are using those in the code above.

In this exercise, you will implement a class `Appointment` that stores the date and description of an appointment:

```
public class Appointment
{
   private Date date;
   private String description;
   private DateFormat formatter = DateFormat.getDateTimeInstance(
         DateFormat.MEDIUM, DateFormat.SHORT);

   . . .

   public String toString()
   {
      return "Appointment[date="
           + formatter.format(date)
           + "description=" + description + "; "
           + "]";
   }
}
```

Supply a constructor

```
Appointment(String aDate, String aDescription)
```

that constructs an `Appointment` object from two strings. If the first string is not in a legal format, your constructor should throw a `ParseException`.

Part 2

Now you will create a class `AppointmentBook` that keeps an array list of `Appointment`s. Part of the class has been provided for you:

```
import java.util.ArrayList;
import java.text.ParseException;

public class AppointmentBook
{
   private ArrayList<Appointment> book;

   public AppointmentBook()
   {
      book = new ArrayList<Appointment>();
   }

   . . .

   public void addAll(ArrayList<Appointment> list)
   {
      book.addAll(list);
   }

   public int getNumAppointments()
   {
      return book.size();
   }

   public Appointment getAppointment(int i)
   {
      return book.get(i);
   }

   public String toString()
   {
      String out = "";
      for (Appointment a : book)
      {
         out = out + a.toString() + "\n";
      }
      return out;
   }
}
```

Add a method `add`, that adds a new appointment to the book. The method should not catch the `ParseException`, but propagate it to the calling method (throw it up the call stack).

```
public void add(String aDate, String aDescription)
```

Part 3

Write a program `AppointmentBookDemo` whose `main` method asks the user to enter a series of appointments. Add the appointments into an appointment book object. If a parse error occurs in the `add` method, have the program instruct the user to reenter the appointment.

Here is an outline of the `main` method, which does not yet catch any exceptions.

```
boolean done = false;
while (!done)
{
   System.out.println("Next date (or -1 when done):");
   String input1 = in.nextLine();

   if (input1.equals("-1"))
      done = true;
   else
   {
      System.out.println("Description:");
      String input2 = in.nextLine();
      book.add(input1, input2);
   }
}
System.out.println(book);
```

Add the appropriate `try/catch` block and test your program.

What is the complete code for your `AppointmentBookDemo` class?

Part 4

The `AppointmentBookDemo` program is tedious to use because the user must type all appointment data at the prompt. Improve the method so that the data can be stored in a file.

Put the data in a text file, say, `appts.txt`, and provide this file name when the program asks you to.

An `AppointmentBookReader` class will read appointments stored in a file. Use the following class in your solution:

```
import java.util.Scanner;
import java.text.ParseException;
import java.io.FileReader;
import java.io.IOException;

public class AppointmentBookReader
{
   private AppointmentBook book;
```

```
   public AppointmentBookReader()
   {
      book = new AppointmentBook();
   }

   public AppointmentBook read(String filename)
      throws IOException, ParseException
   {
      FileReader reader = new FileReader(filename);
      Scanner in = new Scanner(reader);
      while (in.hasNextLine())
      {
         String input1 = in.nextLine();
         String input2 = in.nextLine();
         book.add(input1, input2);
         reader.close();
      }
      return book;
   }
}
```

To read from an appointment book, use the following code:

```
AppointmentBookReader bookReader = new AppointmentBookReader();
AppointmentBook book = bookReader.read(filename);
System.out.println(book);
```

Now copy the `AppointmentBookDemo` program to a new class -- maybe, `AppointmentBookDemo2` -- so that it uses an `AppointmentBookReader`.

If there is any exception, describe the nature of the exception in English. Print the contents of the appointment book at the end of your program, whether or not there is any exception.

What is the complete code for your `AppointmentBookDemo2` class?

Part 5

Give an example of an input file that causes the `AppointmentBookReader` to throw a `ParseException`.

Part 6

Write an `AppointmentBookWriter` class that provides methods for saving an AppointmentBook to a text file. Write a new Driver class that opens a data file with an `AppointmentBookReader` and prints out the contents. Then it allows the user to enter new appointments (like in Part 3), and finally saves all appointments back to the data file using `AppointmentBookWriter`. Running this program multiple times should show the address book growing.