Notes for all homework assignments

Due date and time:
Due at 11:55pm, Saturday, Mar 7, 2015.

Tools:
Eclipse Classic 4.4 (Luna) on Java SE7 (JDK 7).

Reference materials:
Lecture notes, sample code, and any other references.

Submission:
Make sure your application works in the way you expect. Convert your project folder to a JAR file and then upload it to the Moodle submission box. It can be found right under the link for the homework instruction sheet on Moodle.

Grading policy:
Your homework assignments will be graded on the quality of implementation: code with no errors, code organization by proper indentation and spacing, and code documentation by comments. I will make every attempt to standardize grading policies, but I reserve the right to normalize grades to account for possible disparities. Specifically, the grading policy for each problem is as follows:

- **(0 credit)** If there is any unhandled/unfixed syntax error in the solution.
- **(10% off for each logic error)** A logic error is an error that, in general, produces wrong output just because of wrong algorithm or logic (e.g., wrong specification of precedence on mathematical operations) or improper use of API. A logic error causes either the program to halt or the program to return wrong results if the program containing the logic error runs.
- **(0 credit)** If the solution you turn in, even without any error in it, is not substantial to be deemed a reasonable effort.
- **(90% of the max credit)** If the solution you turn in runs without generating any type of error and produce correct results.
- **(10% of the max credit)** This portion of credit is based on code organization by proper indentation, spacing, and code documentation by comments. To get the credit, you program should first run without errors and produce correct results.
- **(5% off the max credit per instruction not followed)** The minimum requirement for the homework assignments is that you need to stick to the instructions given for them.
- **(Bonus credit up to 10%)** You are strongly encouraged to *algorithmically* enhance your solutions as much as you can. Note that the starting point for any enhancement is the instructions given for the assignments. This means that you should at least fulfill the requirements set by the instructions and then you may go beyond the minimum requirements.

IMPORTANT: Put your name as a comment at the top of each class file (i.e., *xyz*.java). Also use appropriate package names (e.g., hw1p1, hw1p2, etc.) unless otherwise specified. You may test each of your classes for a problem via a main method. However, your solution to each problem should have only one main method, regardless of the number of classes created for each problem.

Problem #1 (20 points): Date, Calendar, GregorianCalendar, final blank class and instance variables; static initializer; constructors.

1. Java classes related to date and calendar.
    a. The class java.util.Date represents a specific instant in time, with millisecond precision.
    b. The java.util.Calendar class is an abstract class that provides methods for converting between a specific instant in time and a set of calendar fields such as YEAR, MONTH, DAY_OF_MONTH, HOUR, and so on, and for manipulating the calendar fields, such as getting the date of the next week.
        i. An abstract class is a class that can't be instantiated. For example, Calendar c = new Calendar() would not work.
        ii. However, you can create a reference variable of an abstract class and assign an instance of its subclass (i.e., widening is allowed). For example, Calendar c = new GregorianCalendar() is legal.
    c. The class java.util.GregorianCalendar is a concrete subclass of Calendar and provides the standard calendar system used by most of the world.
        i. A concrete class is a class that can be instantiated. We have been working on concrete classes.
        ii. For example,
        ```
        Calendar gcal = new GregorianCalendar(); // widening
        gcal.set(GregorianCalendar.DAY_OF_WEEK, Calendar.FRIDAY);
        // 1 for first week, 2 for second week…, -1 for last week
        gcal.set(GregorianCalendar.DAY_OF_WEEK_IN_MONTH, -1);
        ```
    d. You can get the current date and time from the platform as follows:
    Calendar c = Calendar.getInstance(); or
    Date date = gcal.getTime();
2. Create a Java project named HW4-*FirstName-LastName* (e.g., HW4-Seung-Lee).
    a. Add a new package named hw4p1 to the project.
    b. Add four classes to the package: MyCalendarTester, LastSundayTester, MyCalendar, and LastSunday. As usual, do not include main method in the last two classes.
3. For the MyCalendar class, do the following:
    a. Declare a *blank* final class variable named NUMBER_OF_DAYS. Use static initializer to assign 365 if the given year is a common year. For a leap year, assign 366.
    b. Declare five blank final instance variables named CURRENT_YEAR, CURRENT_MONTH, CURRENT_DAY, CURRENT_HOUR, and CURRENT_MINUTE.
        i. Write a default constructor to assign values to the variables. Hints: Start with Calendar calendar = Calendar.getInstance();

ii. Write an instance constructor with the parameter date of type Date. Hints: Begin with Calendar gcalendar = new GregorianCalendar();
iii. Write another instance constructor with the parameter stringDate of type String. Hints: Begin with DateFormat and SimpleDateFormat classes as follows:
DateFormat dateFormat = new SimpleDateFormat("MMMM d, yyyy HH:mm", Locale.ENGLISH);
Date date = dateFormat.parse(stringDate);
Calendar gc = new GregorianCalendar();

You are going to pass in a date string argument when you instantiate the class MyCalendar using this instance constructor. For example, MyCalendar mc = new MyCalendar("January 10, 1970 09:12");

Note that when you use the parse method of DateFormat class, it could throw an ParseException exception (i.e., the parse method is a "dangerous" method that could generates an error. In this case, the parse method will generate an error when the string argument passed in cannot be parsed because of some reason such as a wrong string content. We will get into this later. When a dangerous method is called in another method, the method signature for the calling method must include throws ParseException.

Thus the signature for the third constructor looks like
public MyCalendar(String stringDate) throws ParseException

4. For MyCalendarTester class, do the following:
    a. Create three instances of the MyCalendar class by the different constructors.
    b. Print the date in the format: *monthName day*, *year hh:mm* (e.g. February 21, 2015 14:32).
5. For the LastSunday class, do the following:
    a. Write three overloaded methods named printLastSunday with empty parameter list. The method should print the information about the last Sunday of last week for the date. It also print the date as it is and as a formatted date as well. See an example output below:

    The date is Fri Feb 27 14:19:33 CST 2015
    The formatted date is February 21 2015
    The last Friday of last week in February: Fri Feb 27 14:19:33 CST 2015

    Hints: Begin with Calendar gcal = new GregorianCalendar(). Use a date format specifier. Also use the set and getDisplayName methof of GregoryCalendar class.
6. For the LastSundayTester class, do the following:
    a. Instantiate the LastSunday class.
    b. Call the printLastSunday method to produce the output that looks like the one above.

Problem #2 (10 points): Wrapper class, methods of the String class, etc.

1. Create a package named hw4p2.
2. Right click on the hw4p2 package of the project and add classes named IntegerConversion, DoubleConversion, HexConversion, OctalConversion, and ConversionTester.
3. Write code in a method for each of the conversion classes.
    a. If it is integer conversion, get a string from a user. The string should be a series of integers, each separated by a comma and a space (e.g., 11, 2, 36, 4, 8, 7, 3, 256).
    b. Convert the string into the series of integers, add the numbers together, and print the result.
    c. Repeat the instructions a and b above for other classes.
    d. Consider using an array, methods of String class (e.g., split(), contains(), startWith(), substring()), methods of Double wrapper class (e.g., parseDouble()), and methods of the Integer wrapper class (e.g., parseInt()).
4. For the main method of ConversionTester class.
    a. The main method should get the user's choice via a dialog. Depending on the user's choice, call the method from the relevant class.
    b. To create a dialog taking an input, use the following statement:

        JOptionPane.showInputDialog(null, "Type 1 for integer conversion, 2 for double conversion, etc.: ", "Conversion Options", JOptionPane.QUESTION_MESSAGE);

        Consider \n for the long prompt.

        where the second string is a prompt, the third one is for a title of the dialog box, and the last one is a constant defined by the JOptionPane class (You need to import this class; use Ctrl + Shift + o in Eclipse to import a class with its package).

    c. You must handle the case where the user does not choose an option. That is, you must display a message when a user does not choose an option through the dialog box. The input you take via the dialog box should be cast into an integer using Integer.parseInt(). The parseInt method throws NumberFormatException. You should catch this exception. Use try…catch block (see here).
        i. If this is not null, you should display a message box by the statement: JOptionPane.showMessageDialog(null, "You didn't type an option!"). If there is no error, prompt the user to provide a series of integers and print them as converted. Use the JOptionPane again to get the series of integers.

```
try{
      //statements
} catch(NumberFormatException ex) {
      System.out.println(ex.getMessage());
}
```

Problem #3 (10 points): 2D ragged array

5. Create a package named hw4p3.
6. Add two classes named TwoDimenArray and Launcher to the package.
7. In the RaggedArray class, write the following methods:
    a. public static int max(int[][] arr) that returns the maximum value in the 2D parameter array arr.
    b. public static int rowColumnSum(int[][] arr, int x) that returns the sum of the elements in row x and column x of arr. Caution: Be careful with rows of different lengths!).
8. Declare the following arrays in the main method of the Launcher class and test the methods you have written in the TwoDimenArray class.
    c. int[][] sqrs = { {2,4,6}, {8,10,12}, {14,16,18} }; // not ragged
    d. int[][] negatives = { {-11,-9,-3}, {-4,-5,-6,-8}, {-7,-8} }; // ragged
    e. int[][] nonsqrs = { {1,2,3}, {4,5}, {6,7,8,9} }; // ragged

Problem #4 (10 points): 2D normal array

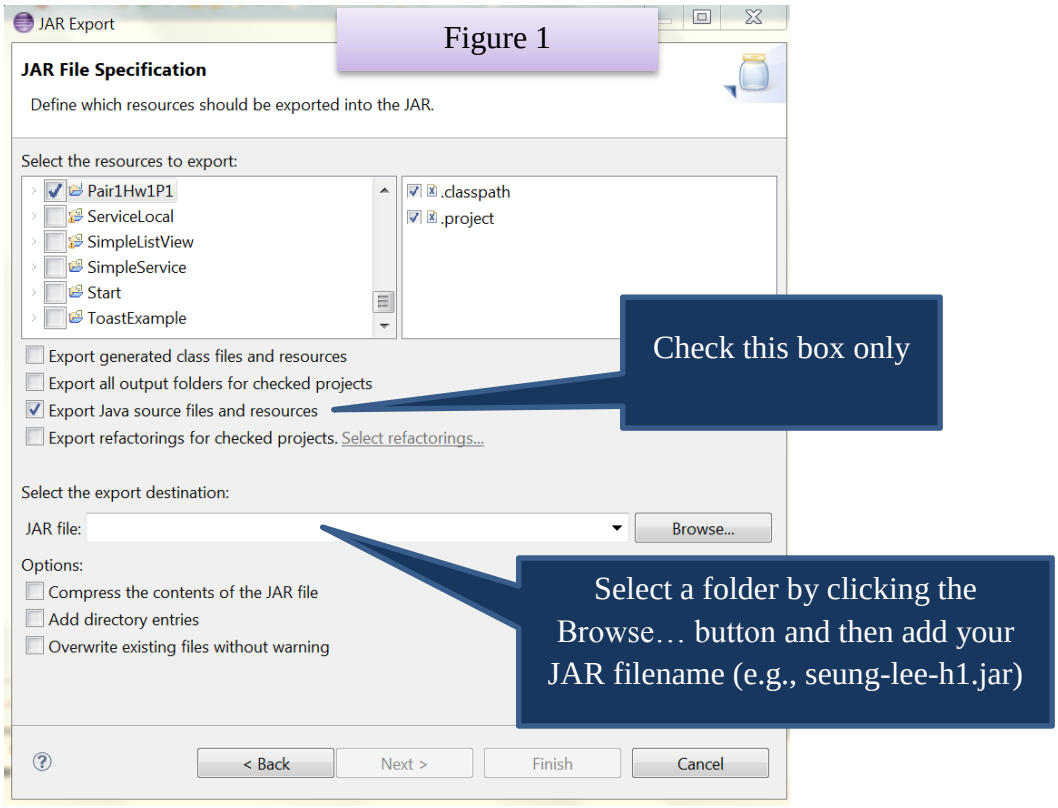
1. Create a Java package named hw4p4.
2. Right click on the hw4p3 package of the project and add two classes Customer and Launcher.
3. For the Customer class:
    a. Declare private fields: name, item, and amount (This means order amount). A customer can order only one item per transaction.
    b. Write getters and setters for the name and item fields.
        i. The name should not exceed 6 characters.
        ii. The item must be one of these: ipad, iphone, and macbook. Assume that the prices of the items are 5, 10, and 15 dollars, respectively.
    c. Write only a getter for the amount field.
4. For the Launcher class:
    a. Instantiate 3 customers: John, Mary, and Jane
    b. Declare a 2D String array with 6 orders: 2 by John, two by Mary, and two by Jane.
    c. Each order (i.e., row) consists of name, item, and amount.
    d. Print the order details and with totals.

How to submit the homework assignment?

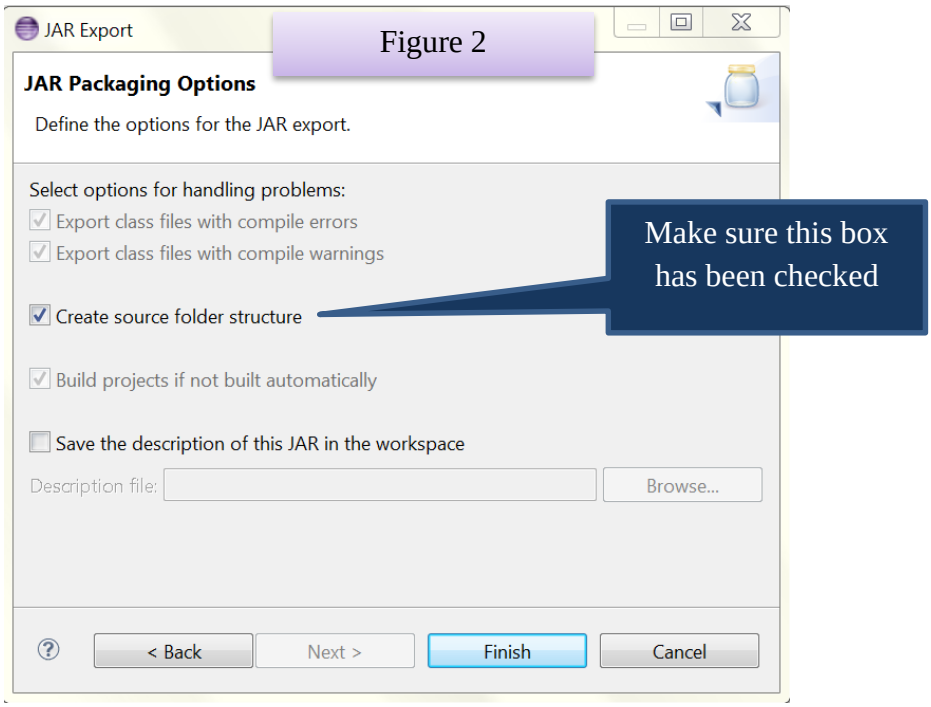
1. Note: If you do not create a JAR file as directed, you will lose 3 points
2. When you create a JAR (Java ARchive) file (.jar) for your project, the name of the JAR file must match the project name for grading purpose. To do this follow the steps below:
    a. Right click on the project (e.g., HW1-Seung-Lee) in the **Package Explorer** of Eclipse and then click **Export…**.
    b. Expand the **Java** node and select **JAR file**. Click **Next**.
    c. In the **JAR Export** window (see Figure 1 below), make sure you check the items as shown in the image below: ***your project name***, **.classpath**, **.project**, and **Export Java source**

files and resources, and then provide a path to the **JAR file** textbox. The path must include the JAR file name (e.g., C:\Users\Seung\Documents\EclipseWorkspace\Jar*HW1-Seung-Lee.jar*).

i. Click the **Browse…** button for the **Select the export destination** textbox.
ii. Find and open the folder where you want to store the JAR file.
iii. Type your JAR filename. It must be the same as your project name + the extension .jar (e.g., HW1-Seung-Lee.jar). Note that the JAR filename could be different from the project name but for grading purpose you must use your project name as the JAR filename.
iv. Click the **Save** button.
v. You will go back to the **JAR Export** window, now with the **Next** button available.
vi. Click the **Next** button.



d. In the next window (see Figure 2 below), tick the **Create the source folder structure** checkbox if it has not already been checked and then click the **Finish** button.



3. Upload the JAR file (e.g. HW1-Seung-Lee.jar).
    a. Go to Moodle and find out the link for the current homework instruction.
    b. You can find another link (with an icon) right under the homework instruction link with the link text **HW4**.
    c. Click the link and upload your JAR file only.
    d. Note that once you upload your JAR file, you cannot reverse or overwrite it.