Test 2 Review 1

CompSci 6

Fall 2005

Name (print): ________________________________

Honor Acknowledgment (signature): ____________________________________________

DO NOT SPEND MORE THAN 15 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later. The final page is a list of common methods of classes we have studied in class so that you do not need to memorize such details.

Before starting, make sure your test contains 13 pages.

If you think there is a syntax error, then ask.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>8 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>60 pts.</td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 1:  *(Cutting through the confusion: (16 points))*

Many web pages embed parameters to their web pages within the page’s URL (Uniform Resource Locator). This is often done with scripted pages such as you would find at online stores. This information appears after the web page’s name and is encoded such that each piece is separated from other pieces by an ampersand, &. A piece is made up of an id and a value which are separated by an equal sign, =.

Several examples are given below. The first line below has three pieces. The first piece has the id *name* and the value *harry*, the middle piece has the id *type* and the value *wizard*, and the last (or third) piece has the id *diet* and the value *potions*.

"name=harry&type=wizard&diet=potions"
"name=ethan&year=1999&month=june&color=blue",
"name=robert&animal=tiger",
"name=penny&type=penguin&zoo=asheboro",
"name=denny&type=duke&weight=120",

**Part A: (5 points)**

Write a method `pieceCount` that returns the number of pieces in a URL.

For example the code below sets `x` to 1 and `y` to 5.

```java
int x = pieceCount("name=marco");
int y = pieceCount("name=mo&type=zebra&weight=360&height=71&zoo=bronx");
```

Complete the method, `pieceCount`, below.

```java
/**
 * Return the number of pieces (as defined above) in url
 */
public int pieceCount (String url)
{
```

}  

```
**Part B:** (5 points)

Write a method `getNthPiece` that returns just the specified piece from the given URL.
For example, the code below sets first to "name=moe" and last to "zoo=bronx"

```java
string zoostuff = "name=moe&type=zebra&weight=360&height=71&zoo=bronx";
string first = getNthPiece(zoostuff, 1);
string last = getNthPiece(zoostuff, 5);
```

Complete the method, `getNthPiece`, below.

```java
/**
 * Return the n-th piece of url (where n is guaranteed to be
 * between 1 and pieceCount(url), inclusive).
 */
public String getNthPiece (String url, int n) {
```
**Part C**: (6 points)

Write a method `splitURL` that splits a given URL into two parallel lists, one containing the `ids` and one the `values`.

For example, the following URL would fill the list `ids` with the strings “name”, “year”, “month”, and “color” and the list `values` with the strings “ethan”, “1999”, “june”, and “pink” respectively. In this way, the second element of the list `ids` corresponds to the second element of the list `values`:

```
"name=ethan&year=1999&month=june&color=pink"
```

You will not receive full credit for this part unless you call the methods `pieceCount` and `getNthPiece` that you wrote previously and use their results in determining the result of this function. Assume each works as specified regardless of what you wrote.

Complete the method, `splitURL`, below.

```java
/**
 * Splits each piece of given url such that ids contains all
 * of the id fields in url and values contains all of the
 * value fields in url. You may assume that both lists are
 * non-null and simply add the pieces to the end of each list.
 */
public void splitURL (String url, List<String> ids, List<String> values) {
```
PROBLEM 2:  (We know all about sets: (8 pts))

Write the method isSubset below that returns true if the parameter first is a subset of second and false otherwise.

A set $A$ is said to be a subset of $B$ if and only if every element of $A$ is also an element of $B$. For example if

$$A = \{ \text{Apple, Grapefruit, Orange, Lemon} \},$$
$$B = \{ \text{Orange, Tangarine, Lemon, Lime, Grapefruit, Pummelo} \}, \text{ and}$$
$$C = \{ \text{Apple, Orange, Tangarine, Lemon, Lime, Grapefruit, Pummelo} \}$$

Then, $A$ is not a subset of $B$ (because Apple is not in $B$), but both $A$ and $B$ are subsets of $C$. $C$ is not a subset of $A$ or $B$, but all sets are subsets of themselves, so $C$ is a subset of $C$.

You may not make any assumptions about the types of the objects contained in either of the given sets. You will not receive full credit if you cast the objects in the given sets to any type more specific than Object.

Complete the method, isSubset, below

```java
/**
 * Returns true is first is a subset of second (i.e., all items in
 * first are contained within second).
 */
public boolean isSubset (Set<Object> first, Set<Object> second)
{
...
}
```
PROBLEM 3: \( \text{Iterator, I hardly know her: (16 points)} \)

Write a class \texttt{DigitIterator} that is designed to facilitate accessing each digit in a number sequentially from the least significant, i.e., last, digit to the most significant, i.e., first, digit using the familiar set of iterator functions instead of the mathematical operations \texttt{mod} and \texttt{div} directly.

For example, the code below represents how the class might be used

```java
DigitIterator iter = new DigitIterator(987654);
while (iter.hasNext())
{
    System.out.print(iter.next() + " ");
}
```

to print the digits in the opposite order they appear in the number:

\[
\begin{array}{cccccc}
4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\textbf{Part A: (6 points)}

Using the \texttt{DigitIterator} class, write a function \texttt{digitSum} that returns the sum of the digits in a number. For example, the call \texttt{digitSum(1075)} returns \(1 + 0 + 7 + 5 = 13\). Also, \texttt{digitSum(81)} returns \(9\) since \(8 + 1 = 9\).

Use the following code to convert an \texttt{Integer} object, returned by the \texttt{DigitIterator}'s \texttt{next} method to an int for computing with:

```java
int val = (iter.next()).intValue();
```

Complete the method \texttt{digitSum} below.

```java
public int digitSum (int num)
{
    DigitIterator iter = new DigitIterator(num);
```
**Parts B and C:** (10 points)
Complete the declaration of the class’ instance variables and methods below. You should add any private state (data variables) you think your DigitIterator class needs in order to implement the constructor and other two methods of the class so that they function as shown in the example above.

Note, you may lose points for instance variables that are declared there, but not used in any method’s implementation.

Complete the class declaration below.

```java
public class DigitIterator
{
    // Part B: declare instance variables here

    // Part C: implement methods
    public DigitIterator (int number)
    {
    }

    public boolean hasNext ()
    {
        // TODO: returns true only if there are more digits in number
    }

    public Integer next ()
    {
        // TODO: advances to the next digit and returns current digit
    }
}
```
PROBLEM 4:  \((\text{Happy Birthday: (20 points)})\)

Part A: (9 points)
Write a several methods for a class that represents a date as three integer values, one for the day, one for the month, and one for the year:

1. a constructor that takes a string representing the month, day, and year, each value separated by a forward slash, "/", e.g., "7/27/2004"

2. a method that compares the current date to another by first comparing their year, then, if those were equal, their month, and finally, if they were equal, their day; returning 1 if the current date is greater (comes after) the given date, 0 if they represent the same day, or -1 if it is less (comes before) the given date

3. a method that determines if the current date is the same as another by verifying that both have the same month, day, and year

```java
public class Date implements Comparable {
    private int myDay;
    private int myMonth;
    private int myYear;

    public Date (String line) {
```
/**
 * Returns 1 if this date represents a day that is later than the given date,
 *      0 if this date represents the same day as the given date, and
 *     -1 if this date represents a day that is earlier than the given date.
 */
public int compareTo (Date other) {

}

/**
 * Returns true if this date represents the same day as the given date,
 *     false otherwise
 */
public boolean equals (Date other) {

}
Part B: (5 points)

Given a file of dates, one date per line, write a method `readDates` that, given a `Scanner` set to read that file, constructs and returns a List of `Date` objects represented by each line in the file.

Each line of the file has a date in the format of three integer values representing the month, day, year, respectively, each separated by a forward slash, “/”. The dates given are in no particular order.

For example, the following file represents three dates.

```
8/6/2004
7/13/2003
7/27/2004
```

Complete the method below; you should not assume that any additional instance variables exist.

```java
private List<Date> readDates (Scanner input)
{
    // Your code here
}
```
**Part C:** (6 points)

Write a method `closestWithoutGoingOver` that, given a single `Date` and a list of `Date` objects, returns the date in the list that is the fewest days from the target date but not after the given date. If there is no such date, then the target date should be returned.

For example, given the date 7/27/2004 and the following list of dates:

```
7/25/2004
1/10/2004
2/2/2005
7/28/2004
5/15/2004
8/2/2004
```

Your method, `closestWithoutGoingOver` should return 7/25/2004 because it is the closest date to 7/27/2004 that is not in the future.

You can assume that the list dates given contains at least one item.

```java
public Date closestWithoutGoingOver (Date target, List<Date> dates) {
```
Throughout this test, assume that the following classes and methods are available. These classes are taken directly from the material used in class. There should be no methods you have never seen before here. However, if you know of additional methods not listed here, you may use any of them in your solutions.

### String

```java
public class String
{
    // Returns the length of this string.
    public int length()

    // Returns a new string that is a substring
    // of this string. The substring begins at
    // the specified beginIndex and extends to
    // the character at index endIndex - 1.
    public String substring (int beginIndex, int endIndex)

    // Returns the index within this string of
    // the first occurrence of str
    public int index0 (String str)

    // Returns the index within this string of the
    // first occurrence of str after index start
    public int index0 (String str, int start)

    // Returns the index within this string of
    // the last occurrence of str
    public int lastIndex0 (String str)
}
```

### Integer

```java
public class Integer
{
    // The smallest value of type int
    public static final int MIN_VALUE

    // The largest value of type int
    public static final int MAX_VALUE

    // Returns the integer represented by the
    // argument as a decimal integer.
    public static int parseInt (String s)

    // Returns a new String object representing
    // the specified integer.
    public static String toString (int i)

    // Returns value of Integer object as an int
    public int intValue ()
}
```

### Iterator

```java
public interface Iterator
{
    // Returns true if iteration has more elements.
    boolean hasNext ()

    // Returns the next element in this iteration.
    Object next ()
}
```

### Object

```java
public class Object
{
    // Returns true iff o is the same as this object
    boolean equals (Object o)

    // Returns string representation of this object
    String toString ()
}
```

### TreeSet

```java
public class TreeSet implements Set
{
    // Constructs a new, empty set
    public TreeSet()

    // Returns an iterator over the elements in
    // this set. The elements are returned in
    // ascending order.
    public Iterator iterator()

    // Returns the number of elements in this set.
    public int size()

    // Returns true if this set contains o
    public boolean contains (Object o)

    // Adds the specified element to this set
    // if it is not already present.
    public boolean add (Object o)

    // Adds the specified collection of objects to
    // this set if they are not already present.
    public booleanaddAll (Collection c)
}
```
ArrayList

class ArrayList implements List
{
    // Constructs an empty list
    public ArrayList()

    // Returns the number of elements in this list.
    public int size()

    // Searches for the first occurrence of the given
    // argument, returns -1 if not found
    public int indexOf(Object item)

    // Returns element at index in this list.
    public Object get(int index)

    // Replaces the element at the specified position
    // in this list with the specified element.
    public Object set(int index, Object element)

    // Appends specified element to end of this list.
    public boolean add(Object o)

    // Appends all elements from c to end of this list.
    public boolean add(Object c)

    // Returns an iterator over the elements in
    // this list. The elements are returned in
    // the order they were added.
    public Iterator iterator()
}

Scanner

class Scanner implements Iterator
{
    // Create Scanner that reads data from a file.
    public Scanner(File file)

    // Create Scanner that reads data from a string.
    public Scanner(String str)

    // Change delimiters used to separate items
    public void useDelimiter(String characters)

    // Check if more items are available
    public boolean hasNext()

    // Get next delimited item as a string
    public String next()

    // Get next delimited item as an integer value
    public int nextInt()
}