Midterm 3 Solution

CS 1323, Spring 2017

Name (printed legibly): ______________________________

Student number: ______________________________

Integrity Pledge

On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature: ____

Do not write anything on the back of any page. If you need extra space use the blank page at the back of the examination. If you pull the test apart, please put <u>all</u> of the pages in order when you hand your exam in.

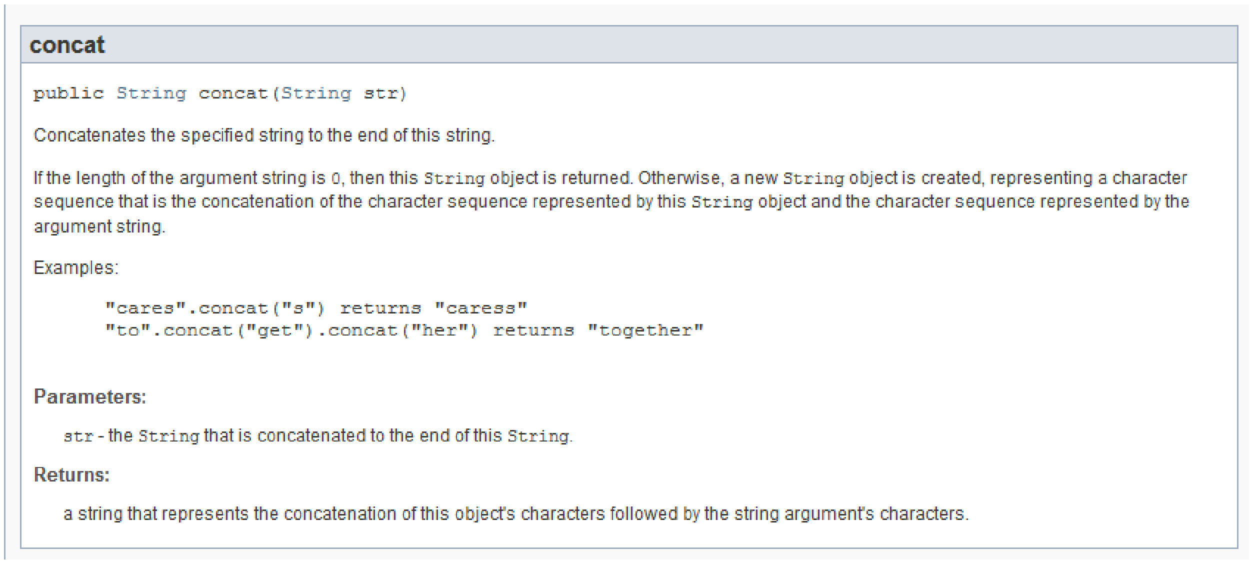
Answer all programming questions in Java. Show your work to receive partial credit.

Pay careful attention to what is requested: a code fragment (a few lines of code), a method, or a program.

Since you do not have the whole API available during the examination, it is acceptable to guess at method names, parameters and return values. If your guesses are reasonable—even if not perfect—you will receive credit. If, however, you make up new methods that are not reasonable for the class or that magically solve problems the class cannot solve, you will not get credit.

You do not need to import packages or throw exceptions in any methods.

Relevant parts of the API are given in a separate document.



concat

```
public String concat(String str)
```

Concatenates the specified string to the end of this string.

If the length of the argument string is `0`, then this `String` object is returned. Otherwise, a new `String` object is created, representing a character sequence that is the concatenation of the character sequence represented by this `String` object and the character sequence represented by the argument string.

Examples:

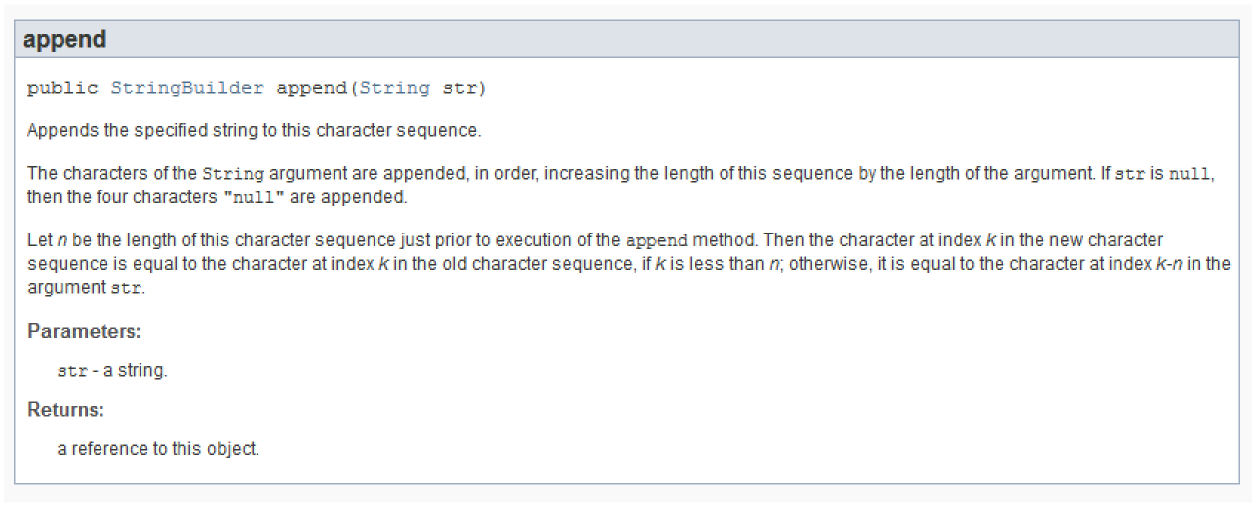
```
      "cares".concat("s") returns "caress"
      "to".concat("get").concat("her") returns "together"
```

Parameters:

`str` - the `String` that is concatenated to the end of this `String`.

Returns:

a string that represents the concatenation of this object's characters followed by the string argument's characters.



append

```
public StringBuilder append(String str)
```

Appends the specified string to this character sequence.

The characters of the `String` argument are appended, in order, increasing the length of this sequence by the length of the argument. If `str` is `null`, then the four characters `"null"` are appended.

Let *n* be the length of this character sequence just prior to execution of the `append` method. Then the character at index *k* in the new character sequence is equal to the character at index *k* in the old character sequence, if *k* is less than *n*; otherwise, it is equal to the character at index *k-n* in the argument `str`.

Parameters:

`str` - a string.

Returns:

a reference to this object.

Do not write anything on the back of any page

1. (10 points; 2 points each for a) and b); 6 points for c))

a) What does the String object day contain after the following code is executed?

```
String day = new String("Monday ");
day = day.concat("Blues"); // see API on back of previous pag
```

Monday Blues

b) What does the StringBuilder object day contain after the following code is executed?

```
StringBuilder day = new StringBuilder("Friday ");
day.append("Fun"); // see API on back of previous page
```

Friday Fun

c) Write a method returns "Found it!" if a given target String is in a given ArrayList and "Not there!" otherwise. If the ArrayList contained {"Today", "Tomorrow", "Yesterday"} and the target was "Today", "Found it!" should be returned. If the target were "Next Week", "Not there!" should be returned.

```
public static String contains(ArrayList<String> data, String target)
{
      for (int index = 0; index < data.size(); ++index)
      {
            if (data.get(index).equals(target))
            {
                  return "Found it!";

            }
      }

      return "Not there!";

}
```

Do not write anything on the back of any page

2. (10 points; 3 points for algorithm, 7 for sorting) Sort the values below using either insertion sort or selection sort. You must indicate which algorithm you used and show the process exactly as it was in class***, with each individual swap or move of data shown on a separate line***. **Only data that are moved or changed should be shown.**

I used (circle one): selection sort insertion sort

Selection sort

2	4	6	5	3	1	Aux
1					2	
	2				4	
		3		6		
			4		5	
				5	6	

Insertion sort

2	4	6	5	3	1	Aux
						4
	4					
						6
		6				
						5
			6			
		5				
						3
				6		
			5			
		4				
	3					
						1
					6	
				5		
			4			
		3				
	2					
1						

3. (10 points) Trace the code fragment below in the table below.

```
int[] data = {9, 4, 2, 1, 5};

for (int stop = data.length; stop > 0; --stop)
{
      for (int index = 0; index < stop-1; ++index)
      {
            if (data[index] > data[index+1])
            {
                  int temp = data[index];
                  data[index] = data[index+1];
                  data[index+1] = temp;
            }
      }
}
```

stop	index	temp	contents of data after if statement
5	0	9	4, 9, 2, 1, 5
	1	9	4, 2, 9, 1, 5
	2	9	4, 2, 1, 9, 5
	3	9	4, 2, 1, 5, 9
	4		
4	0	4	2, 4, 1, 5, 9
	1	4	2, 1, 4, 5, 9
	2		
	3		
3	0	2	1, 2, 4, 5, 9
	1		
	2		
2	0		
	1		
1	0		
0			

4. (10 points; 2 points each for a) to d), 4 points for e)) Use the documentation for the Dictionary class (given in a separate document) to answer the following questions and perform the following tasks.

In parts a) to c), if there are no methods of the required type, say “None.”

a) List one constructor in the Dictionary class

Dictionary()

b) List one mutator method in the Dictionary class.

put(), remove()

c) List one accessor method in the Dictionary class.

getAllKeys(), getAllValues(), getValue(), isEmpty(), size()

d) Are Dictionary objects mutable or immutable?

mutable

e) Create a Dictionary object that contains the following key-value pairs: (“final”, “a value that cannot be changed”), (“while”, “repeats actions”), (“return”, “end the method”).

```
Dictionary dict = new Dictionary();

dict.put(“final”, “a value that cannot be changed”);

dict.put(“while”, “repeats actions”);

dict.put(“return”, “end the method”);
```

5. (40 points; 10 points for a), 15 points for b) and 15 points for c))

The U.S. government has a no-fly list that holds the identities of people who are not permitted to board aircraft in the United States. Since people's names are not unique, a passport number is stored. Passport numbers contain 8 digits and are followed by USA. 12345678USA could be a passport number.

In this problem you will write the software that manages the no-fly list. Your software will read in the no-fly list from a file. It will then allow only valid passport numbers that are not already in this list to be added to the list. When the program ends, it should write the current no-fly list out to a file.

The passport numbers must be stored in an oversized array, kept in sorted order so binary search can be used. You may assume that there are no more than 10000 passports on the list. The signatures of the methods are below. ***You only have to implement three methods.*** You may use other methods in your solution without implementing them.

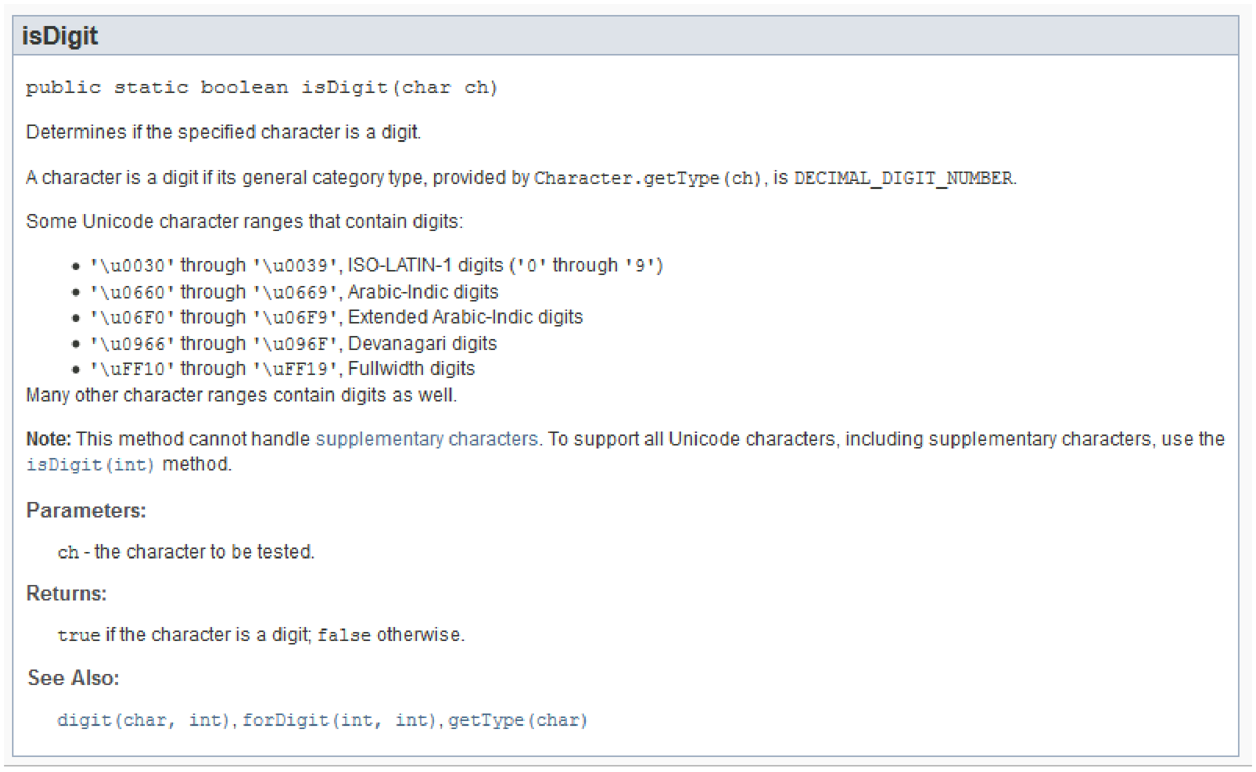
```
public static boolean isValid(String passport)
public static int add(String[] passports, int passportSize, String insertMe)
public static boolean search(String[] passports, int passportSize, String target)
public static int readFile(String[] passports, String fileName)
public static void writeFile(String[] passports, int passportSize, String fileName)
```

Suppose the no-fly list initially contains:

```
11111117USA
12345678USA
22222224USA
33333331USA
44444448USA
55555555USA
```

A sample run of the program is below.

```
Enter 1 to add, 2 to search, or 3 to quit
1
Enter the passport number
66666662USA
Enter 1 to add, 2 to search, or 3 to quit
2
Enter the passport number
66666662USA
That person may not fly.
Enter 1 to add, 2 to search, or 3 to quit
1
Enter the passport number
123123123USA
That passport is not valid
Enter 1 to add, 2 to search, or 3 to quit
1
Enter the passport number
A2312314USA
That passport is not valid
Enter 1 to add, 2 to search, or 3 to quit
3
```



isDigit

```
public static boolean isDigit(char ch)
```

Determines if the specified character is a digit.

A character is a digit if its general category type, provided by `Character.getType(ch)`, is `DECIMAL_DIGIT_NUMBER`.

Some Unicode character ranges that contain digits:

- `'\u0030'` through `'\u0039'`, ISO-LATIN-1 digits (`'0'` through `'9'`)
- `'\u0660'` through `'\u0669'`, Arabic-Indic digits
- `'\u06F0'` through `'\u06F9'`, Extended Arabic-Indic digits
- `'\u0966'` through `'\u096F'`, Devanagari digits
- `'\uFF10'` through `'\uFF19'`, Fullwidth digits

Many other character ranges contain digits as well.

Note: This method cannot handle supplementary characters. To support all Unicode characters, including supplementary characters, use the `isDigit(int)` method.

Parameters:

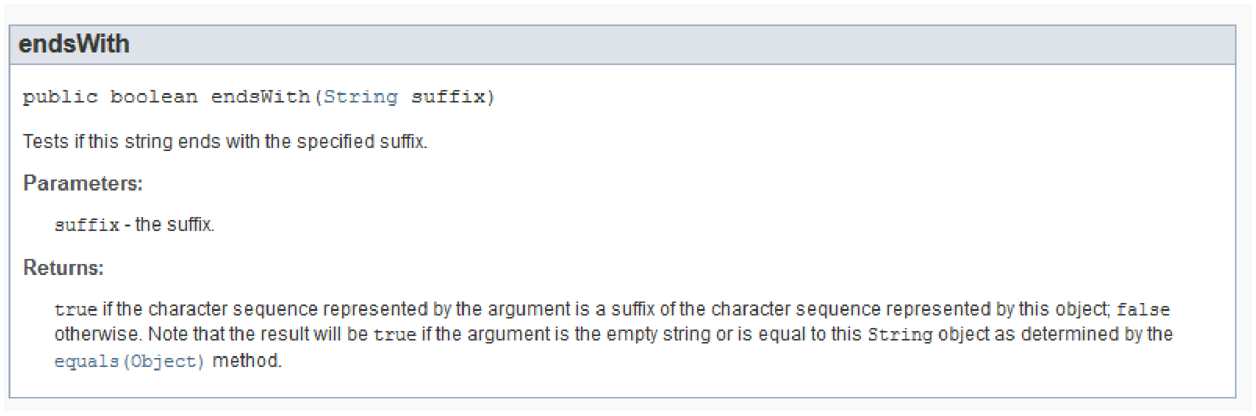
`ch` - the character to be tested.

Returns:

`true` if the character is a digit; `false` otherwise.

See Also:

`digit(char, int)`, `forDigit(int, int)`, `getType(char)`



endsWith

```
public boolean endsWith(String suffix)
```

Tests if this string ends with the specified suffix.

Parameters:

`suffix` - the suffix.

Returns:

`true` if the character sequence represented by the argument is a suffix of the character sequence represented by this object; `false` otherwise. Note that the result will be `true` if the argument is the empty string or is equal to this `String` object as determined by the `equals(Object)` method.

a) Implement the isValid method below. There are three ways that a passport number can be invalid. The passport number might not start with eight digits. The passport number might not end with “USA”. The passport number may not have eleven characters. ***Use constants in your code.***

There are methods in the Character and String classes that will be useful. The API for these methods is on the back of the previous page.

```
public static boolean isValid(String passport)
{
      final int DIGITS = 8;
      final int TOTAL_LETTERS = 11;

      if (!passport.endsWith("USA"))
            return false;

      if (passport.length() != TOTAL_LETTERS)
            return false;

      // Check for only digits in first 8 spots
      for(int index = 0; index<DIGITS; ++index)
      {
            char c = passport.charAt(index);
            if (!Character.isDigit(c))
                  return false;
      }

      return true;
}
```

b) Implement the add method. ***The array must be kept in sorted order.***

```
public static int add(String[] passports, int passportSize, String insertMe)
{
      // binary search will be used both to find if the data is in the
      // array already and to guide insertion
      int insertion = Arrays.binarySearch(passports, 0,
                              passportSize, insertMe);

      // Check to see that the passport number is not already in array
      if (insertion >= 0)
      {
            System.out.println("That person is already in the array");
            return passportSize; // already in array--our work is finished
      }

      // insertion is negative--find where the value should be inserted
      int place = -insertion - 1;
      int index;
      for (index = passportSize; index > place; --index)
      {
            passports[index] = passports[index-1];
      }
      passports[index] = insertMe;

      return passportSize + 1;
}
```

Programming notes: It is possible to write this method by inserting in the last position of the list and sorting, but that is inefficient. The method used here is inspired by insertion sort.

c) Complete the main program below using only the methods described. You may not modify the existing code. Javadoc for the methods in the class is in a separate document.

```
// main method signature
public static void main(String[] args)
{
      String inputFile = "NoFly4-23-17.txt";
      String outputFile = "NoFly4-24-17.txt";

      Scanner input = new Scanner(System.in);

      //TODO Create the array and read in the file
      final int SIZE = 10;
      String[] passports = new String[SIZE];
      int passportSize = 0;

      passportSize = readFile(passports, inputFile);

      // Priming read
      System.out.println("Enter 1 to add, 2 to search, or 3 to quit");
      int choice = input.nextInt();
      input.nextLine();

      while (choice != 3)
      {
            System.out.println("Enter the passport number");
            String ppt = input.nextLine();

            //TODO Check to see that ppt is a valid passport
            if (!isValid(ppt))
                  System.out.println("That passport is not valid");

            else if (choice == 1)
            {
                  //TODO Add ppt to the array
                  passportSize = add(passports, passportSize, ppt);
            }
            else if (choice == 2)
            {
                  //TODO Search the array for ppt
                  if (search(passports, passportSize, ppt))
                  {
                        System.out.println("That person may not fly.");
                  }
                  else
                  {
                        System.out.println("Happy flying!");
                  }
            }

            // Priming read
            System.out.println("Enter 1 to add, 2 to search, or 3 to quit");
            choice = input.nextInt();
            input.nextLine();
      }
```

```
        // TODO Write out the file
        writeFile(passports, passportSize, outputFile);
        input.close();
}
```