Midterm 3 Solution

CS 1323, Fall 2015

Name (printed legibly):

Student number:

Section (1 meets at 3 p.m., 10 meets at 9:30 a.m., 995 is on the internet):

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 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, unless asked to do so.

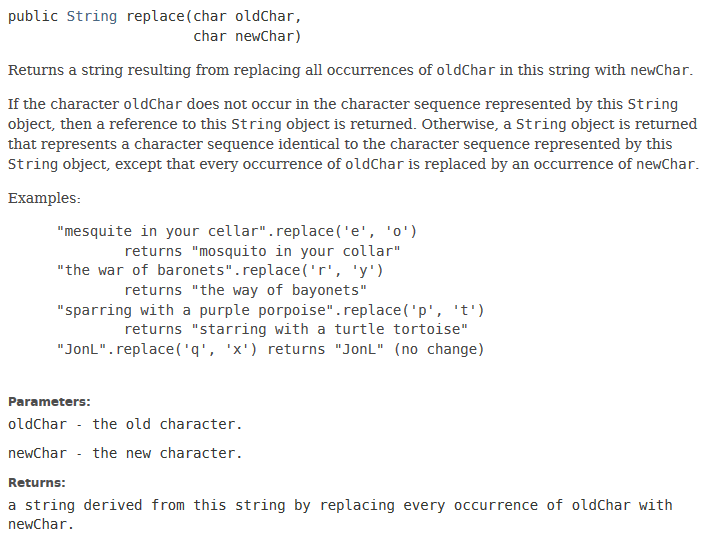
Relevant parts of the API are copied on to the reverse side of pages.

From the String class:

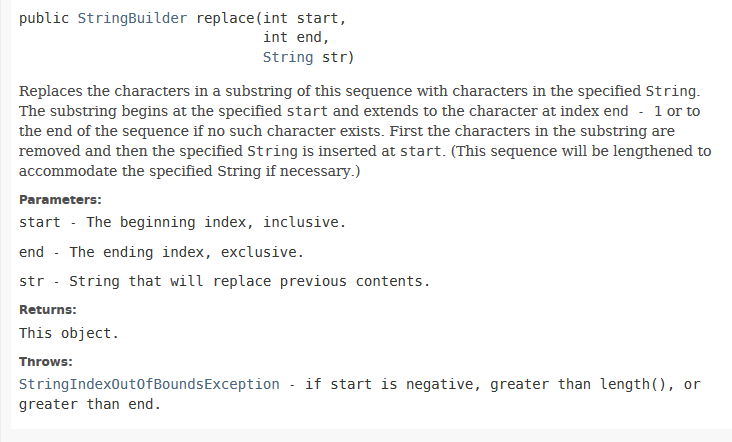
Integrity Pledge

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

Signature (or initials if taking the exam through ProctorU):



From the StringBuilder class



1. (20 points; 4 points each)

a) What does the StringBuilder object word contain after the following code is executed?

StringBuilder word = new StringBuilder(“Turkey”);

word.replace(0, 3, “Milark”); // see documentation on back of previous page

"Milarkkey"

b) What does the String object word contain after the following code is executed?

String word = new String(“Turkey”);

word.replace(‘k’,’n’); // see documentation on back of previous page

"Turkey" (Strings are immutable)

c) Suppose that there were a method in the StringBuilder class that is similar to append (that inserts at the end of a StringBuilder) but inserts at the beginning. This method could be called appendStart. For each method signature below, determine whether the method is a class method (static) or an instance method and ***circle the appropriate answer***

void appendStart(StringBuilder sb, String prefix)

class method

void appendStart(String prefix)

instance method

d) What does list contain after the following sequence of calls?

ArrayList<Integer> list = new ArrayList<Integer>();

list.add(new Integer(3)); list.add(new Integer(5)); list.add(new Integer(7));

ArrayList<Integer> remove = new ArrayList<Integer>();

remove.add(new Integer(3)); remove.add(new Integer(6));

list.removeAll(remove); // see documentation on back of this page

5 and 7

e) What will be returned by the method below?

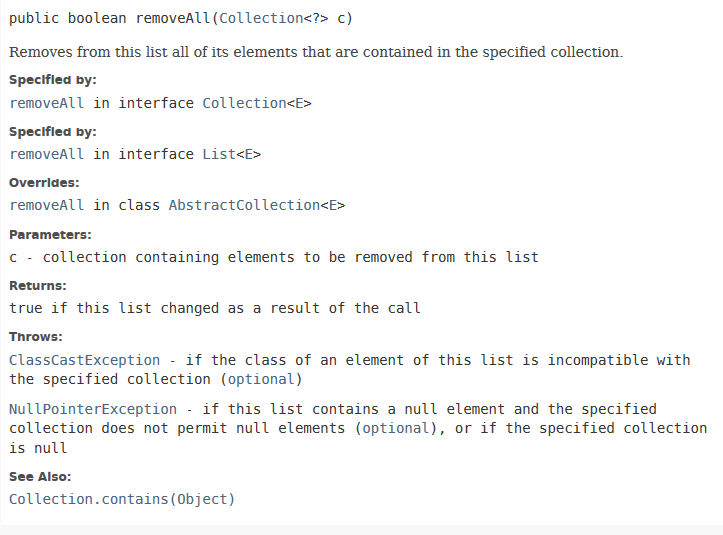
ArrayList<Integer> list = new ArrayList<Integer>();

list.add(new Integer(5)); list.add(new Integer(3)); list.add(new Integer(7)); // Pay attention to order!

Collections.binarySearch(list, new Integer(5));

Garbage since the data was not sorted.

Method for the ArrayList class



1. (25 points; 5 points each) The paragraphs below describe static methods. Determine the signature of the method **ONLY**. The signature of a method is the return type, the method name, and the parameters. ***Do not write the method(s).***

a) The method takes an ArrayList<String> that holds a dictionary of properly spelled words (one to a line) and returns whether or not a given word is in the dictionary.

For example: If the dictionary contained {“hello”, “world”}, then true would be returned if “hello” is sought, and false would be returned if “goodbye” was sought.

boolean findWord(ArrayList<String> list, String word)

b) The method takes two definitions that are stored as String objects and determines whether they have any words in common or not.

For example: If the first definition was “an instance is an object” and the second definition was “an object is an instance”, this method would return true. If the first definition was “an instance is an object” and the second definition was “methods perform actions”, this method would return false.

boolean common(String first, String second)

c) The method takes two definitions and determines which, if any, words they have any words in common.

For example: If the first definition was “an instance is an object” and the second definition was “an object is an instance”, this method would return an ArrayList containing {“an”, “object”, “is”, “an”, “instance”}. If the first definition was “an instance is an object” and the second definition was “methods perform actions”, this method would return an ArrayList that is constructed but contains no elements.

ArrayList<String> common (String first, String second)

d) The method takes an ArrayList<String> and a String object and adds the String to the ArrayList if the given String object is not already in the ArrayList<String>.

For example: If the ArrayList<String> contained {“a”, “b”} and the given String was “c”, the ArrayList would contain {“a,”, “b”, “c”} after the method call. If the given String had been “a”, the ArrayList<String> would not be changed.

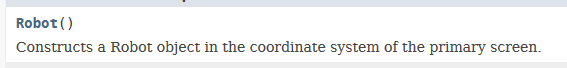
void addWord(ArrayList<String dictionary>, String word)

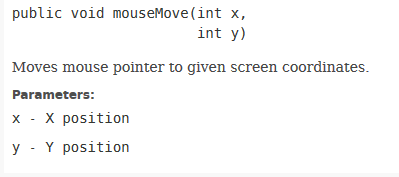
e) The method takes two ArrayList<String> objects and returns an ArrayList<String> object that contains the Strings that are common to both of the original objects.

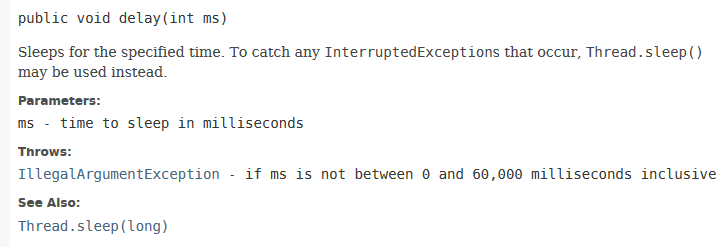
For example: If the first ArayList<String> contained {“a”, “b”, “c”} and the second contained {“a”, “b”, “d”} the result would be an ArrayList<String> that contains {“a”, “b”}. If the second had contained {“d”, “e”} the resulting ArrayList<String> would be empty.

ArrayList<String> common(ArrayList<String> first, ArrayList<String> second)

Some parts of the Robot Class

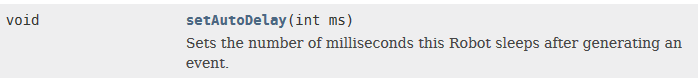
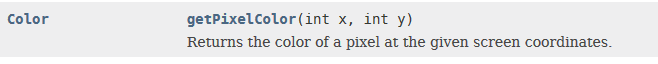












1. (25 points; 4 points for a), and b); 2 points for c) and d); 6 points for e) and f); 1 point free of charge))

***Use the documentation for part of the Robot class (given on the back of the previous page) to answer the following questions and perform the following tasks.***

Robot is a cool Java class that artificially imitates a user doing things like pressing keys on the keyboard and moving the mouse around. It is used to test computer programs without forcing some poor person to try to repeat sequences of actions. Since humans move much slower than computers, there is a standard amount of time that is delayed between actions (delays are all given in milliseconds—which are 1/1000 of a second).

a) List two accessor methods in the Robot class. If there are none, say “None.”

getAutoDelay

getPixelColor

b) List two mutator methods in the Robot class. If there are none, say “None.”

mouseMove

mousePress

delay

setAutoDelay

c) Based on your answers to a) and b), is this class mutable or immutable?

Mutable (although answer will depend on a) and b)

d) Construct a Robot object.

Robot robot = new Robot();

e) Given a Robot object with reference robot, write a code fragment that moves the mouse across the screen to (100, 200) after a 500 millisecond time delay.

robot.delay(500);

robot,mouseMove(100, 200);

f) Different mice have different numbers and types of buttons to click. There are int constants in the InputEvent class for three mouse buttons called BUTTON1\_MASK (left), BUTTON2\_MASK (middle), and BUTTON3\_MASK (right) that are used to tell Java which mouse button to click. Write the code that presses the middle button on the Robot object robot.

robot.mousePress(InputEvent.BUTTON2\_MASK);

1. (30 points; 15 points each for a) and b)

You work for a bricks and mortar automobile dealership that must regularly perform a manual inventory where you count the number and types of cars that are still on the lot. The dealership uses a special format to collect this data. Each car is described in the following way:

Model #Doors Color Price ID#

For example:

Accord 4 White 23498.99 111222333

This is a White 4 door Accord that costs $23498.99, and has an ID# of 111222333. You may assume the ID has 9 integral digits (and can therefore be stored as an int).

As always, we assume that the people who input the data never make any mistakes.

Your program needs to input the data in the given format, and analyze and print the data as follows:

1. All car summaries in the format given above
2. Number of cars on the lot
3. Total cost of cars on the lot
4. Number of cars by model

The data will be stored in an ArrayList<String>.

***The following class methods are available for you to use (DO NOT WRITE THESE METHODS***). Each of the methods takes a single String in the format above and returns one part of the String. For example, the getModel() method would take “Accord 4 White 23498.99 111222333” and return “Accord”. The getDoors(String line) method would take “Accord 4 White 23498.99 111222333” and return 4.

public static String getModel(String line)

public static int getDoors(String line)

public static String getColor(String line)

public static double getPrice(String line)

public static int getID(String line)

1. Write a method that gets the input from the user. This method should prompt the user to enter data, get input from the user, store each data element entered in an ArrayList<String>. The signature of the method is below. The Scanner parameter will be passed an argument initialized to the keyboard of the computer. The prompt contains the String that should be used to prompt the user. The flag parameter will contain the sentinel the user enters to tell the program that he or she is finished entering data.

public static ArrayList<String> enterData(Scanner keyboard, String prompt, String flag)

{

ArrayList<String> result = new ArrayList<String>();

System.out.println(prompt);

String input = keyboard.nextLine();

while (! input.equals(flag))

{

result.add(input);

System.out.println(prompt);

input = keyboard.nextLine();

}

return result;

}

1. Write a method that prints out the number of cars by model. The best way to do this is to sort the ArrayList alphabetically. Since the model is given first, this will put the ArrayList in order by model. You then need to step through the ArrayList one element at a time to perform the task. Do not try to use binarySearch to perform this task.

There is a method in the problem description that will be useful and necessary.

public static void printCarsByModel(ArrayList<String> carList)

{

Collections.sort(carList);

String prev = carList.get(0);

String prevModel = getModel(prev);

int count = 1;

for (int i=0; i<carList.size(); ++i)

{

String current = carList.get(i);

String currentModel = getModel(current);

if (prevModel.equals(currentModel))

{

++count;

}

else

{

System.out.println(prevModel + ": " + count);

count = 1;

prevModel = currentModel;

}

} // end for

}