1、2019 真题——Delimiters

Many encoded strings contain delimiters. A delimiter is a non-empty string that acts as a boundary between different parts of a larger string. The delimiters involved in this question occur in pairs that must be balanced, with each pair having an open delimiter and a close delimiter. There will be only one type of delimiter for each string. The following are examples of delimiters.

Example 1

Expressions in mathematics use open parentheses "(" and close parentheses ")" as delimiters. For each open parenthesis, there must be a matching close parenthesis.

(x + y) \* 5 is a valid mathematical expression.

(x + (y) is NOT a valid mathematical expression because there are more open delimiters than close delimiters.

Example 2

HTML uses <B> and </B> as delimiters. For each open delimiter <B>, there must be a matching close delimiter </B>.

<B> Make this text bold </B> is valid HTML

<B> Make this text bold </UB> is NOT valid HTML because there is one open delimiter and no matching close delimiter.

In this question, you will write two methods in the following Delimiters class.

public class Delimiters

{

/\*\* The open and close delimiters. \*/

private String openDel;

private String closeDel;

/\*\* Constructs a Delimiters object where open is the open delimiter and close is the

\*  close delimiter.

\*  Precondition: open and close are non-empty strings.

\*/

public Delimiters(String open, String close)

{

openDel = open;

closeDel = close;

}

/\*\* Returns an ArrayList of delimiters from the array tokens, as described in part \*(a). \*/

public ArrayList<String> getDelimitersList(String[] tokens)

{  /\* to be implemented in part (a) \*/  }

/\*\* Returns true if the delimiters are balanced and false otherwise, as described in \*part (b).

\*  Precondition: delimiters contains only valid open and close delimiters.

\*/

public boolean isBalanced(ArrayList<String> delimiters)

{  /\* to be implemented in part (b) \*/  }

// There may be instance variables, constructors, and methods that are not shown.

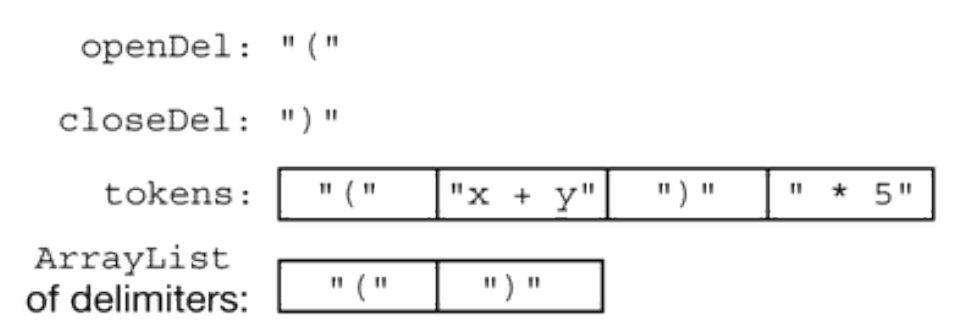
}

(a)

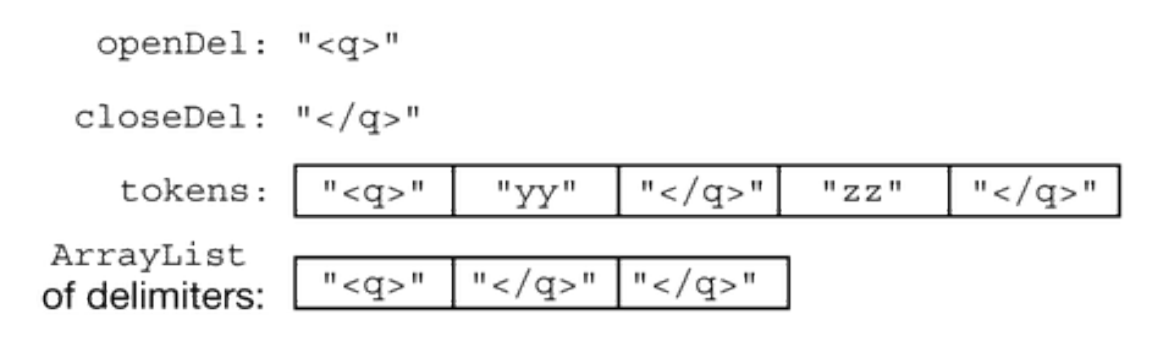
A string containing text and possibly delimiters has been split into tokens and stored in String[] tokens. Each token is either an open delimiter, a close delimiter, or a substring that is not a delimiter. You will write the method getDelimitersList, which returns an ArrayList containing all the open and close delimiters found in tokens in their original order.

The following examples show the contents of an ArrayList returned by getDelimitersList for different open and close delimiters and different tokens arrays.

Example 1



Example 2



Complete method getDelimitersList below.

/\*\* Returns an ArrayList of delimiters from the array tokens, as

described in part (a). \*/

public ArrayList<String> getDelimitersList(String[] tokens)

(b)

Write the method isBalanced, which returns true when the delimiters are balanced and returns false

otherwise. The delimiters are balanced when both of the following conditions are satisfied; otherwise, they are not

balanced.

1. When traversing the ArrayList from the first element to the last element, there is no point at which

there are more close delimiters than open delimiters at or before that point.

2. The total number of open delimiters is equal to the total number of close delimiters.

Consider a Delimiters object for which openDel is "<sup>" and closeDel is "</sup>". The examples

below show different ArrayList objects that could be returned by calls to getDelimitersList and the value

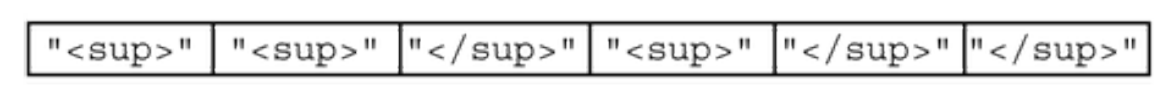
that would be returned by a call to isBalanced.

Example 1

The following example shows an ArrayList for which isBalanced returns true. As tokens are examined

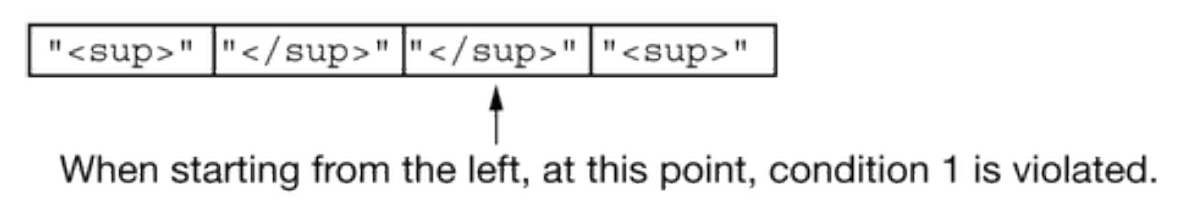
from first to last, the number of open delimiters is always greater than or equal to the number of close delimiters. After

examining all tokens, there are an equal number of open and close delimiters.



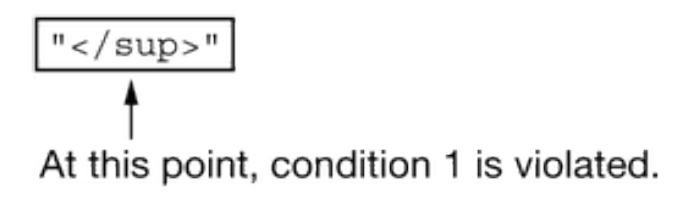
Example 2

The following example shows an ArrayList for which isBalanced returns false



Example 3

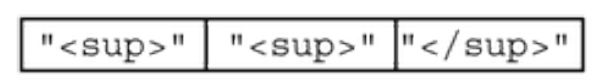
The following example shows an ArrayList for which isBalanced returns false.



Example 4

The following example shows an ArrayList for which isBalanced returns false because the second

condition is violated. After examining all tokens, there are not an equal number of open and close delimiters.



Complete method isBalanced below.

/\*\* Returns true if the delimiters are balanced and false otherwise, as

described in part (b).

\* Precondition: Precondition: delimiters contains only valid open and close

delimiters.

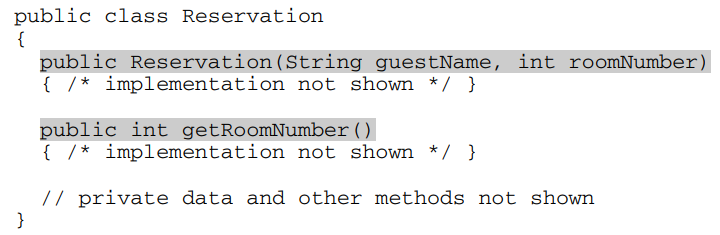
\*/

public boolean isBalanced(ArrayList<String> delimiters)

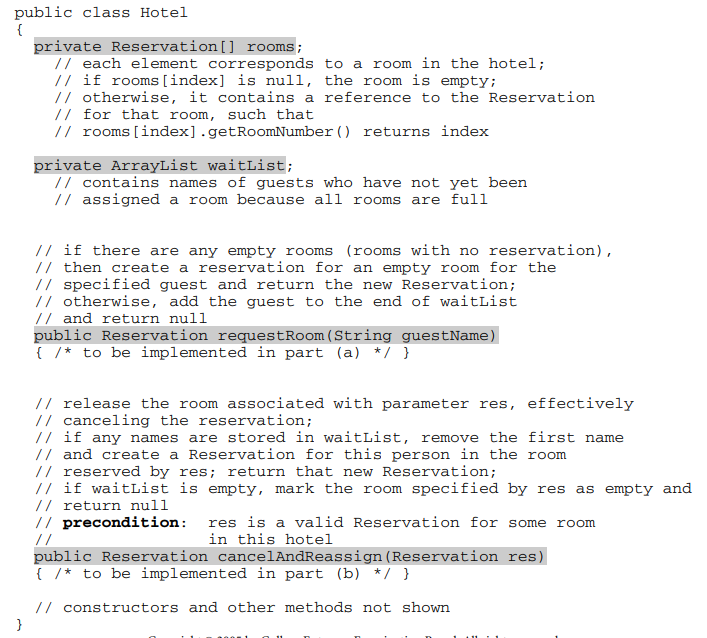
2、2005 真题 Hotel reservations

In this question, you will implement two methods for a class Hotel that is part of a hotel reservation system.

The Hotel class uses the Reservation class shown below. A Reservation is for the person and room number specified when the Reservation is constructed.

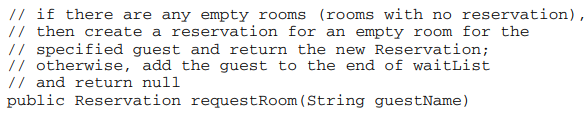


An incomplete declaration for the Hotel class is shown below. Each hotel in the hotel reservation system has rooms numbered 0, 1, 2, . . . , up to the last room number in the hotel. For example, a hotel with 100 rooms would have rooms numbered 0, 1, 2, . . . , 99.



1. .Write the Hotel method requestRoom. Method requestRoom attempts to reserve a room in the hotel for a given guest. If there are any empty rooms in the hotel, one of them will be assigned to the named guest and the newly created reservation is returned. If there are no empty rooms, the guest is added to the end of the waiting list and null is returned.

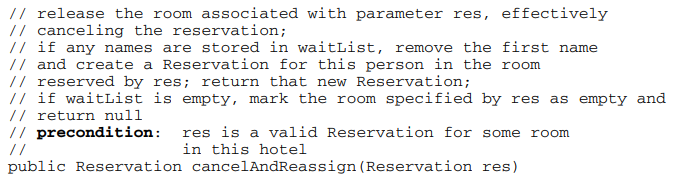
Complete method requestRoom below.



b). Write the Hotel method cancelAndReassign. Method cancelAndReassign releases a previous reservation.

If the waiting list for the hotel contains any names, the vacated room is reassigned to the first person at the beginning of the list. That person is then removed from the waiting list and the newly created reservation is returned. If no one is waiting, the room is marked as empty and null is returned. In writing cancelAndReassign you may call any accessible methods in the Reservation and Hotel classes. Assume that these methods work as specified.

Complete method cancelAndReassign below.



1. 考纲案例

The Gizmo class represents gadgets that people purchase. Some Gizmo objects are electronic and others are not. A partial definition of the Gizmo class is shown below.

public class Gizmo

{

/\*\* Returns the name of the manufacturer of this Gizmo. \*/

public String getMaker()

{ /\* implementation not shown \*/ }

/\*\* Returns true if this Gizmo is electronic, and false

\* otherwise.

\*/

public boolean isElectronic()

{ /\* implementation not shown \*/ }

/\*\* Returns true if this Gizmo is equivalent to the Gizmo

\* object represented by the

\* parameter, and false otherwise.

\*/

public boolean equals(Object other)

{ /\* implementation not shown \*/ }

// There may be instance variables, constructors, and methods not shown.

}

The OnlinePurchaseManager class manages a sequence of Gizmo objects that an individual has purchased from an online vendor. You will write two methods of the OnlinePurchaseManager class. A partial definition of the OnlinePurchaseManager class is shown below.

public class OnlinePurchaseManager

{

/\*\* An ArrayList of purchased Gizmo objects,

\* instantiated in the constructor.

\*/

private ArrayList<Gizmo> purchases;

/\*\* Returns the number of purchased Gizmo objects that are electronic

\* whose manufacturer is maker, as described in part (a).

\*/

public int countElectronicsByMaker(String maker)

{ /\* to be implemented in part (a) \*/ }

/\*\* Returns true if any pair of adjacent purchased Gizmo objects are

\* equivalent, and false otherwise, as described in part (b).

\*/

public boolean hasAdjacentEqualPair()

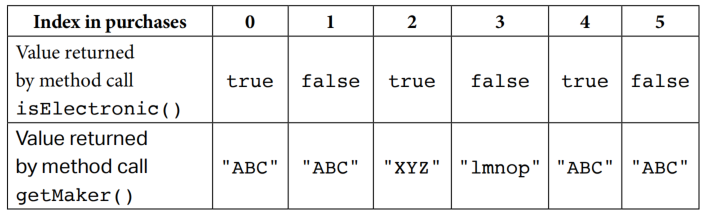
{ /\* to be implemented in part (b) \*/ }

// There may be instance variables, constructors, and methods not shown.

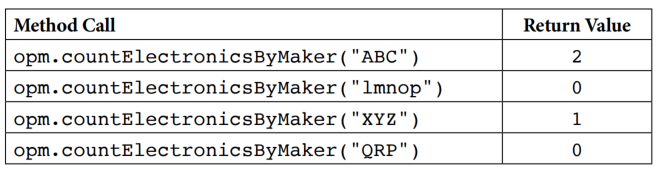
}

(a) Write the countElectronicsByMaker method. The method examines the ArrayList instance variable purchases to determine how many Gizmo objects purchased are electronic and are manufactured by maker.

Assume that the OnlinePurchaseManager object opm has been declared and initialized so that the ArrayList purchases contains Gizmo objects as represented in the following table.



The following table shows the value returned by some calls to countElectronicsByMaker.



Complete method countElectronicsByMaker below.

/\*\* Returns the number of purchased Gizmo objects that are electronic and

\* whose manufacturer is maker, as described in part (a).

\*/

public int countElectronicsByMaker(String maker)

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user’s purchase history to detect such occurrences and request confirmation.

Write the hasAdjacentEqualPair method. The method detects whether two adjacent Gizmo objects in purchases are equivalent, using the equals method of the Gizmo class. If an adjacent equivalent pair is found, the hasAdjacentEqualPair method returns true. If no such pair is found, or if purchases has fewer than two elements, the method returns false.

Complete method below.

/\*\* Returns true if any pair of adjacent purchased Gizmo objects are

\* equivalent, and false otherwise, as described in part (b).

\*/

public boolean hasAdjacentEqualPair()