Mail - Nair, Sharath (Cognizant) | HackerRank Question - Conden

https://www.hackerrank.com/test/9r8p040m0m7/questions/1bk78sqe0

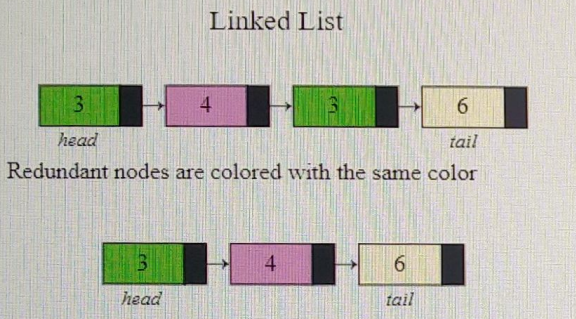
1h 20m left

ALL

1. Condensed List

Given a list of integers, remove any nodes that have values that have previously occurred in the list and return a reference to the head of the list.

For example, the following list has a recurrence of the value *3* initially:



Remove the node at position *2* in the list above, *0* based indexing.

Function Description

Complete the function *condense* in the editor below.

condense has the following parameter(s):

head: the head of a singly-linked list of integers, a LinkedListNode

Note:. A *LinkedListNode* has two attributes: *data*, an integer, and *next*, a reference to the next item in the list or the language equivalent of *null* at the tail.

Returns

reference to LinkedListNode: the head of the list of distinct values

Info Language: Java 8 Autocomplete Ready

```
62
63      /*
64       * Complete the 'condense' function below.
65       *
66       * The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
67       * The function accepts INTEGER_SINGLY_LINKED_LIST head as parameter.
68       */
69
70      /*
71       * For your reference:
72       *
73       * SinglyLinkedListNode {
74       *     int data;
75       *     SinglyLinkedListNode next;
76       * }
77       *
78       */
79
80      public static SinglyLinkedListNode condense(SinglyLinkedListNode head) {
81      // Write your code here
82      HashSet<Integer> hSet = new HashSet<>();
83      SinglyLinkedListNode vtrr1;
84      SinglyLinkedListNode vtrr2;
85      vtrr1 = head;
86      vtrr2= null;
87      while(vtrr1!=null){
88          int val=vtrr1.data;
89          if(hSet.contains(val))vtrr2.next=vtrr1.next;
90          else{
91              hSet.add(val);
92              vtrr2 = vtrr1;
93          }
94          vtrr1=vtrr1.next;
95      }
96      return head;
97
98      }
99
100 }
101
102 > public class Solution {...
```

Test Results Custom Input

equivalent of *null* at the tail.

Returns

reference to LinkedListNode: the head of the list of distinct values

Constraints

- $1 \leq n \leq 10^5$
- $0 \leq LinkedListNode[i].val \leq 1000$

▶ **Input Format for Custom Testing**

▼ **Sample Case 0**

Sample Input 0

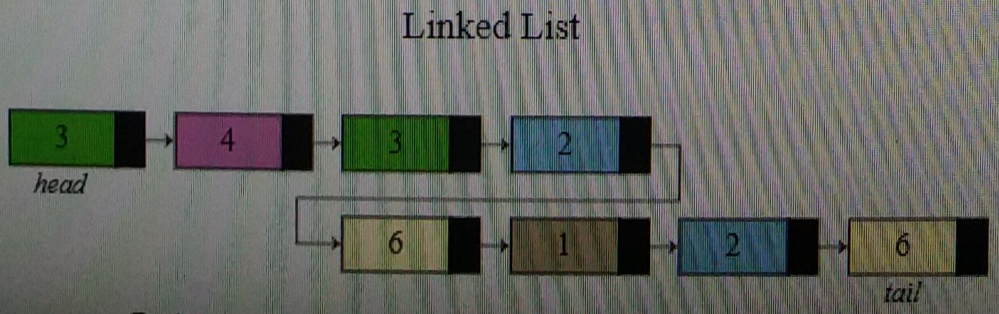
```
STDIN       Function Parameters
-----       -------------------

8       →   list[] Size n = 8
3       →   list[] = [ 3, 4, 3, 2, 6, 1, 2, 6 ]
4
3
2
6
1
2
6
```

Sample Output 0

```
3
4
2
6
1
```

Explanation 0

The list looks like this:



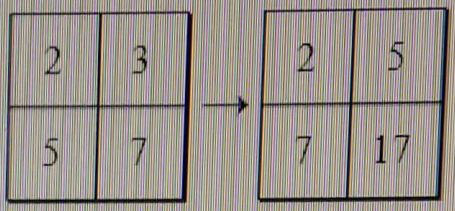
1h 19m left

ALL

2. Matrix Summation

The algorithm below is used to convert the "before matrix" to "after matrix". Find the "before matrix" given the "after matrix".

Algorithm	Example

`s = 0;`
 	
`for (i = 0; i ≤ x; i += 1) {`
`  for (j = 0; j ≤ y; j += 1) {`
`  s = s + before(i, j);`
`  }`
`}`
 	
`after(x, y) = s;`
	
	If array *before = [[2,3], [5,7]]*, then *after = [[2,5], [7,17]]*

The algorithm is run for each *after(x, y)* to determine their values. Given *after*, find the original values in *before*.

Example

after = [[2, 5], [7, 17]]

Calculating the values in *after* from *before*:

- *after[0][0] = before[0][0] = 2*
- *after[0][1] = before[0][0] + before[0][1] = 2 + 3 = 5*
- *after[1][0] = before[0][0] + before[1][0] = 2 + 5 = 7*
- *after[1][1] = before[0][0] + before[0][1] + before[1][0] + before[1][1] = 2 + 3 + 5 + 7 = 17*

Function Description

Complete the function *findBeforeMatrix* in the editor below.

findBeforeMatrix has the following parameter(s):

after[n][m]: an $n \times m$ array of integers

Returns:

int[n][m]: an $n \times m$ array of integers representing the *before* matrix

Constraints

- $1 \leq n, m \leq 10^3$
- $0 \leq after[i][j] \leq 10^3$, where $0 \leq i < n$ and $0 \leq j < m$

▶ **Input Format for Custom Testing**

▼ **Sample Case 0**

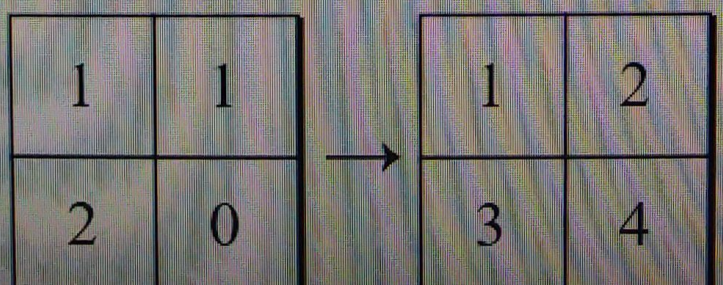
Sample Input

```
STDIN       Function
-----       -----
2       →   size of after[] n = 2
2       →   size of after[][] m = 2
1 2     →   after = [[1, 2], [3, 4]]
3 4
```

Sample Output

```
1 1
2 0
```

Explanation



```java
import java.io.*;

class Result {

    /*
     * Complete the 'findBeforeMatrix' function below.
     *
     * The function is expected to return a 2D_INTEGER_ARRAY.
     * The function accepts 2D_INTEGER_ARRAY after as parameter.
     */

    public static List<List<Integer>> findBeforeMatrix(List<List<Integer>> after) {
    // Write your code here
    int s=0;
    List<List<Integer>> before = new ArrayList<List<Integer>>();
    for(int i=0;i<=after.size()-1;i++){
        List<Integer> bRow = new ArrayList<Integer>();
        for(int j=0;j<=after.get(i).size()-1;j++){
            if(i==0 && j==0){
                bRow.add(j,after.get(0).get(0));
            }
            else if(i==0){
                bRow.add(j,after.get(0).get(j)-after.get(0).get(j-1));
            }else if(j==0){
                bRow.add(j,after.get(i).get(0)-after.get(i-1).get(0));
            }else{
                bRow.add(j,after.get(i).get(j)+after.get(i-1).get(j-1)-after.get(i).get(j-1)-after.get(i-1).get(j));
            }
        }before.add(i,bRow);
    }
    return before;

    }

}

public class Solution {
```

3. Lambda Expressions

```
Line 1: import java.util.*;
Line 2: import java.util.function.*;
Line 3: public class ConsumerDemo {
Line 4:     public static void main(String args[]){
Line 5:     ...
Line 6:      List<Integer> integerList=Arrays.asList(100,200,300,400);
Line 7:      printList(integerList, consumer);
Line 8:   }
Line 9:    public static void printList(List<Integer> listOfIntegers, Consumer<Integer> consumer){
Line 10:    for(Integer integer:listOfIntegers){
Line 11:      consumer.accept(integer);
Line 12:   }
Line 13:  }
Line 14: }
```

The main() method of ConsumerDemo class is expected to print all the values in the integerList. Examine the options given below and select the sta
output?

Pick **ONE** option

- (●) Consumer<Integer> consumer= number -> System.out.print(number +" ");
- ( ) Consumer<Integer> consumer = number -> number;
- ( ) Consumer<Integer> consumer = number -> System.out::print;
- ( ) Consumer<Integer> consumer= consumer.andThen(number ->System.out.println(number + " "));
- ( ) Consumer<Integer> consumer = number -> value;

Clear Selection

4. Streams

```
Take a look at the code given.
Line 1: import java.util.Arrays;
Line 2: import java.util.List;
Line 3: import java.util.stream.*;
Line 4: class SummingClass {
Line 5: public static void main(String args[]){
Line 6:  List<Integer> numbers = Arrays.asList(10,20,30,40);
Line 7:  ...
Line 8:  System.out.println(result);
Line 9:  }
Line 10: }
```

In the above given SummingClass, what possible statement(s) can be inserted at Line 7 to sum all the integers given in the List 'numbers'. Choose FOUR options.

Pick **ONE OR MORE** options

- [ ] int result = numbers.stream().reduce(0, (subtotal)) -> subtotal + element;
- [x] int result = numbers.stream().collect(Collectors.summingInt(Integer::intValue));
- [x] int result = numbers.parallelStream().reduce(0, Integer::sum);
- [x] int result = numbers.stream().reduce(0, Integer::sum);
- [x] int result = numbers.stream().reduce(0, (subtotal, element) -> subtotal + element);

Clear Selection

5. Lambda Expressions

The Sorting class given below sorts the given string array based on the length of the string.

```
Line 1: import java.util.Arrays;
Line 2: import java.util.Comparator;
Line 3: public class Sorting {
Line 4: public static void main(String args[]){
Line 5:  String array[]={"SureshKumar","Ramesh", "Raj"};
Line 6:  Arrays.sort(array, new Comparator<String>(){
Line 7:   public int compare (String str1, String str2){
Line 8:    return(str1.length()-str2.length());
Line 9:   }
Line 10:  });
Line 11: Arrays.stream(array).forEach(str-> System.out.println(str));
Line 12: }
Line 13: }
```

Select the statement(s) from the below options which can replace Line 6 to Line 10 and give the same result. Choose THREE options.

Pick **ONE OR MORE** options

- [x] Arrays.sort(array,(String str1, String str2)->{return(str1.length()- str2.length());});
- [x] Arrays.sort(array,(str1,str2)->{return(str1.length()- str2.length());});
- [x] Arrays.sort(array,(str1,str2)->{int ret = str1.length()-str2.length();return ret;});
- [ ] Arrays.sort(array,(str1,str2))->{return(str1.length()- str2.length());};

Clear Selection

6. Lambda Expressions

Take a look at the code given.

```
Line 1: interface NumberChecker{
Line 2: boolean resolve(int a);
Line 3: }
Line 4: class UtilityClass{
Line 5:    public static boolean checkNumber(NumberChecker numCheck, int number) {
Line 6:        return numCheck.resolve(number);
Line 7:    }
Line 8: public NumberChecker isOddNumber() {
Line 9:      ...
Line 10:   }
Line 11:   public static void main(String args[]){
Line 12:     UtilityClass utility = new UtilityClass();
Line 13:     NumberChecker numCheck = utility.isOddNumber();
Line 14:     System.out.println(numCheck.resolve(5));
Line 15:  }
Line 16:  }
```

To make the UtilityClass print true or false (i.e, odd or not) for the number passed as a parametet at Line 14, select the valid statement(s) that can be inserted Line 9. Choose TWO options

Pick **ONE OR MORE** options

- [x] return odd -> (odd & 1) == 1;
- [ ] return (odd & 1 == 1);
- [x] return odd -> (odd % 2) != 0;
- [ ] return (if (odd & 1) ==1);

Clear Selection

7. Lambda Expressions

```
Take a look at the code given.
Line 1: ...
Line 2: interface Incrementor {
Line 3: ...
Line 4: }
Line 5: class Demo {
Line 6:   public static void main(String args[]) {
Line 7:     Inrementor incrementor = (number) -> number+10;
Line 8:     System.out.println(incrementor.incrementByValue(10));
Line 9:   }
Line 10: }
```

When the above code is compiled and run, it is expected to print the value 20 on the default output stream. What are the statement(s) that need to be inserted at Line 1 and Line 3 to get the expected value.

Pick **ONE OR MORE** options

- [x] Line 1: @FunctionalInterface
  Line 3: public int incrementByValue(int num);
- [x] Line 1: Can be a blank line
  Line 3: public int incrementByValue(int num);
- [ ] Line 1: @FunctionalInterface Line 3: default int incrementByValue(int number) { return number+10; }
- [ ] Line 1: Can be a blank line Line 3: int incrementByValue(int number) { return number+10; }

Clear Selection

Continue

8. Microservice Architecture

Consider a monolithic application that runs all the services together in a single IP address. So whenever there is any downtime in one service, it will create an impact and downtime to the entire application.

Hence to overcome this impact microservices was introduced such that all the services will run independently and will have an impact to the other system.

However as these microservices are running independently, it is not possible to know which services is up/down and where it is running.

Revise this design and provide a solution such that it will solve the above problem
A. Implement Service Discovery Patterns where all the services are registered to the service registry such that the IP address of all the services is known
B. Implement Service Discovery patterns where the services are registered and the ports are exposed via REST APIs
C. Implement Service Discovery patterns where the REST API will support to fetch the IP address of the services are fetched dynamically without any prior registration

Pick **ONE** option

- (●) A,C
- ( ) A,B
- ( ) B,C
- ( ) C only

Clear Selection

9. Microservices

Mr. John is challenged with a discussion on how microservices are prevailing rather than monolithic architecture, while the latter can also be dockerized to avail most of the benefits of micro services. What are the value adds which put microservices on top? Choose TWO Options.

Pick **ONE OR MORE** options

- [x] Less chances of maintaining the fault tolerance.
- [x] Scalability of the applications improves the performance on servers.
- [ ] Size of the application can degrade the performance and slow down the start up.
- [ ] Majorly supports the complex application development.

Clear Selection

10. Microservices

What are the benefit of Single Service per Host model in Microservices? Select THREE options.

Pick **ONE OR MORE** options

- [x] It avoids single point of failure. A failure on one host will only impact one service at a time.
- [x] It makes monitoring and security easier than multiple services on one host.
- [x] It is easier to scale in case we want a specific service to be scaled up.

11. Microservices

Select the benefits of using database per service. Select TWO options.

Pick **ONE OR MORE** options

- [x] It ensure that the services are loosely coupled. Changes to one service's database does not impact any other services.
- [x] Each service can use the type of database that is best suited to its needs.
- [ ] Implementing business transactions that span multiple services is straightforward
- [ ] Implementing queries that join data that is now in multiple databases is easier.

Clear Selection

Clear Selection

12. Spring Boot - Microservices

The train ticket management system allows the customer to book the tickets in various ways like normal booking, takal and premium takal booking. The portal also allows the customer to login and view the booked history or look for available train for various routes. Currently there is a single controller and services of Spring Boot application that takes cares of all the above activities.

As the request and response of this portal is handled by single server and client, there is a performance issue.
Revise this design by choosing one of the below options in such a way that there will be number of services for each request and the performance will be handled by load balancer without affecting the usage of the portal by the customer.

1. Create Eureka Server , which in turn treats itself as a client, where every services is registered to get a dependent microservices to get the job done.
2. Create seperate service class for each use cases of the business need and configure the same in SpringBootApplication class
3. Create multiple REST Client of Spring Boot application to handle the service class

Pick **ONE** option

- ○ I only
- ◉ II only
- ○ Both I and II
- ○ III only

Clear Selection

13. Spring Boot

Consider the given property file (insurance.properties) having the following property details.

policy.lifeInsurance=YEARLY_PREMIUM_OFFER
policy.medicalInsurance=MASTER_HEALTH_CHECKUP_OFFER

From the given options, choose the right definition of PolicyManagement class, that reads from this property file and assigns to its member variable using Spring Boot.

Pick **ONE** option

```
@ConfigurationProperties(prefix="policy")
public class PolicyManagement {
private String lifeInsurance;
private String medicalInsurance;
//getters and Setters for the above
}
```

```
@TypeSafeConfigurationProperty(prefix="policy")
public class PolicyManagement {
private String lifeInsurance;
private String medicalInsurance;

//getters and Setters for the above }
```

```
@ConfigurationProperties
public class PolicyManagement {

@Value{policy.lifeInsurance}
private String lifeInsurance;

@Value{policy.medicalInsurance}
private String medicalInsurance;
```

```
public class PolicyManagement {
private String lifeInsurance;
private String medicalInsurance;

//getters and Setters for the above }
```

```
@ConfigurationProperties
public class PolicyManagement {

@Value{policy.lifeInsurance}
private String lifeInsurance;

@Value{policy.medicalInsurance}
private String medicalInsurance;
}
```

```
@ConfigurationProperties
public class PolicyManagement {

@Value{"$policy.lifeInsurance"}
private String lifeInsurance;

@Value{"$policy.medicalInsurance"}
private String medicalInsurance;
}
```

Clear Selection

14. Spring Boot

What are all the HTTP methods that @RequestMapping can use while you want a Spring Boot class act as a Rest Controller?

Pick **ONE** option

- ○ GET
- ○ POST
- ○ PUT
- ● All of the listed options.

Clear Selection

15. Spring Boot

Shopping Management System allows the customers to order products online. If a customer searches for a product that is not in stock, a custom response, 'Out of stock', along with HTTP_NOT FOUND status should be returned. This exception should be handled using the Spring Boot Exception Handler.
Which of the following code snippets implements the mentioned requirement?

Pick **ONE** option

- (●)
```
@ExceptionHandler(ShoppingManagementException.class)
public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse
exceptionResponse = new ExceptionResponse(exception.getMessage(), "OutOfStock"); return new
ResponseEntity<ExceptionResponse>(exceptionResponse, new HttpHeaders(), HttpStatus.NOT_FOUND);
}
```

- ( )
```
@ExceptionHandler(ShoppingManagementException.class)
public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse
exceptionResponse = new ExceptionResponse(HttpStatus.NOT_FOUND), "OutOfStock"); return new
ResponseEntity<ExceptionResponse>(exceptionResponse);
}
```

- ( )
```
@ExceptionHandler(ShoppingManagementException.class)
public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse
exceptionResponse = new ExceptionResponse(exception.getMessage());
return new ResponseEntity<ExceptionResponse>(exceptionResponse,"OutOfStock", HttpStatus.NOT_FOUND); }
```

- ( )
```
@ExceptionHandler(ShoppingManagementException.class)
public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse
exceptionResponse = new ExceptionResponse(exception.getMessage(), "OutOfStock"); return new
ResponseEntity<ExceptionResponse>( new HttpHeaders(), HttpStatus.NOT_FOUND);
}
```

Clear Selection

16. Spring Boot

Consider a bank management application that maintains all the service related classes in the package com.bank.service.* Choose a code snippet that will load all the beans in the package during the Spring Boot application start up.

Pick **ONE** option

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@Configuration({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@Configuration({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@EnableAutoConfiguration({ "com.bank.controller,com.bank.service,com.bank.dao" })
@Configuration @ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}
```

Clear Selection

Clear Selection

ALL

17. Spring Boot

If you do not want to do default component scan in Spring Boot application, how to disable it?

Pick **ONE** option

- ○ @ComponentScan(disable)
- ◉ @ComponentScan(useDefaultFilters=false)
- ○ @ComponentScan(useDefaultFilter=true)
- ○ None of the listed options.

Clear Selection

18. Spring Boot

Assume in a given scenario, a checked exception occurred in a DAO component and the data flow is in a chain transaction. What will happen to the previously executed transaction?

Pick **ONE** option

18. Spring Boot

Assume in a given scenario, a checked exception occurred in a DAO component and the data flow is in a chain transaction. What will happen to the previously executed transaction?

Pick **ONE** option

- ○ The previous transaction gets rollback even if transaction annotation is not used.
- ◉ The previous transaction do not get rollback automatically even if transaction annotation is used.
- ○ The previous transaction gets rollback even if transaction annotation is used.
- ○ None of the listed options.

Clear Selection

19. Spring Boot

In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given options to implement it. Choose Three options

Pick **ONE OR MORE** options

- ☑ Need to use @EnableAsync in the spring boot application startup class.
- ☑ Need to annotate with @Async in the method from where the other microservice will get invoked .

19. Spring Boot

In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given options to implement it. Choose Three options

Pick **ONE OR MORE** options

- [x] Need to use @EnableAsync in the spring boot application startup class.
- [x] Need to annotate with @Async in the method from where the other microservice will get invoked .
- [ ] Need to have the return type of the method invoking the other microservice as CompletableFuture<T> .
- [x] Need to use ExecuterService class in order to invoke the microservice asynchronously.

Clear Selection

20. Spring Boot

Identify from the given options the file types to be used for a file to hold application specific key/value pairs. Choose TWO options

Pick **ONE OR MORE** options

- [x] application.properties

20. Spring Boot

Identify from the given options the file types to be used for a file to hold application specific key/value pairs. Choose TWO options

Pick **ONE OR MORE** options

- [x] application.properties
- [x] application.yml
- [ ] springboot.properties
- [ ] None of the listed options.

Clear Selection

21. Spring Boot

Select from the given options the way to handle exception globally in Spring Boot application.

Pick **ONE** option

- ○ Using @ExceptionHandler at class level.
- ◉ Using @ControllerAdvice at class level.
- ○ Using @HandlerExceptionResolver at class level.
- ○ None of the listed options.

Clear Selection

22. Spring Boot

You do not want to restart the Spring boot app to reflect the change in configuration file.How to reload the configuration properties without restarting the app? Select your recommendations from the following options.

Pick **ONE** option

- ○ Adding @RefreshConfig at the controller class.
- ○ Adding @RefreshProperties at the controller class.
- ◉ Adding @RefreshScope at the controller class.
- ○ None of the listed options.

Clear Selection

Continue