1. **How can you suggest enhancement to this code snippet.**

**public class ArrArgs {**

**public static void main (String args[]) {**

**try {**

**int k=0;**

**do {**

**System.out.println("Value of input is" + k + "and arguments" + args[k++]);**

**} while (true);**

**} catch (ArrayIndexOutOfBoundsException errorOc) {**

**System.err.println("Eror occured "+errorOc.toString());**

**}**

**}**

**}**

**Answer**

1. An infinite loop is exist need to avoid it by loop iteration limiting to length of args array[].
2. Here can use for loop or for each instead of do While loop.
3. Better to print stack trace for more information instead of error string .

**public class ArrArgs {**

**public static void main (String args[]) {**

**try {**

**int k =0;**

**for (String arg : args) {**

**System.out.println("Value of input is" + k + "and arguments" + args[k++]);**

**}**

**} catch (ArrayIndexOutOfBoundsException errorOc) {**

**System.err.println("Eror occured " + errorOc.toString());**

**errorOc.printStackTrace();**

**}**

**}**

**}**

1. **What is the o/p if java security manager is installed and below program is ran**

import java.util.\*;

import java.io.\*;

public class PermisTest {

public static void main(String[] args) throws Exception {

System.out.println(System.getProperty("user.home"));

try

{

Scanner scObj = new Scanner (new File("input.txt"));

int no1 = scObj.nextInt();

int no2 = scObj.nextInt();

System.out.println("The two nos are : " + no1 + ", " + no2);

// Write to a File

Formatter outObj = new Formatter(new File("output.txt"));

int totalSum = no1 + no2;

System.out.println("The total sum value is " + totalSum);

outObj.format("%d", totalSum);

outObj.close();

}

catch(Exception ee)

{

System.out.println("Error "+ee.toString());

}

}

}

**Answer**

* If security manager has configured to restrict the file access out put will be

**Error: java.security.SecurityException: Access denied**

* If security manager has granted access to files it will read the input file   
   for example input file contain 10 and 12 in first and second line

Will print : **The two nos are : 10, 12**

**The total sum value is 22**

1. **Write a program to occur out of memory error using StringBuffer(while doing append) with storing millions of records and how to resolve it with GC and without GC?**

**Program which cause OutOfMemory Error**

**public class OutOfMemoryErrorGenerator {**

**public static void main(String[] args) {**

**try{**

**StringBuffer sb = new StringBuffer();**

**for (int i = 0; i < Integer.MAX\_VALUE; i++) {**

**sb.append("Record : "+i);**

**}**

**}catch(OutOfMemoryError error){**

**error.printStackTrace();**

**}**

**}**

**}**

**Solution with GC to avoid OutOfMemoryError**

**import java.time.LocalTime;**

**public class ResolveMemoryErrorWithGC {**

**public static void main(String[] args) {**

**LocalTime start = LocalTime.now();**

**try {**

**StringBuffer sb = new StringBuffer();**

**for (int i = 0; i < Integer.MAX\_VALUE; i++) {**

**sb.append("Record : " + i);**

**if(i%1000==0){**

**System.gc();**

**}**

**}**

**} catch (OutOfMemoryError error) {**

**error.printStackTrace();**

**}**

**LocalTime end = LocalTime.now();**

**}**

**}**

**Solution with out GC to avoid OutOfMemoryError**

**import java.io.BufferedWriter;**

**import java.io.File;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**import java.time.LocalTime;**

**import java.util.logging.Level;**

**import java.util.logging.Logger;**

**public class ResolveMemoryErrorWithoutGC {**

**private static final String APPENDING\_VALUE = "Record : ";**

**private static final String STORE\_FILE\_PATH = "C:\\Store\\StoreFile";**

**private static final String EXTENTION = ".txt";**

**public static void main(String[] args) {**

**try {**

**StringBuffer sb = new StringBuffer();**

**int set = 0;**

**for (int i = 1; i < Integer.MAX\_VALUE; i++) {**

**sb.append(APPENDING\_VALUE + i);**

**System.out.println(sb.toString());**

**if (i % 10000 == 0) {**

**set++;**

**writeToFile(sb, set);**

**sb.delete(0, sb.length() - 1);**

**}**

**}**

**} catch (OutOfMemoryError error) {**

**error.printStackTrace();**

**}**

**}**

**private static void writeToFile(StringBuffer sb, int set) {**

**File f = new File(STORE\_FILE\_PATH + set + EXTENTION);**

**try {**

**FileWriter fw = new FileWriter(f);**

**BufferedWriter bw = new BufferedWriter(fw);**

**bw.write(sb.toString());**

**bw.flush();**

**bw.close();**

**} catch (IOException ex) {**

**Logger.getLogger(ResolveMemoryErrorWithoutGC.class.getName()).log(Level.SEVERE, null, ex);**

**}**

**}**

**}**

1. **Write an implementation to read list of items to be added into the restaurant menulist. The items can be sorted naturally based on categories and also explicitly able to sort based on price/name**

**Item Class**

**import java.util.Objects;**

**public class Item {**

**private String category;**

**private String name;**

**private double price;**

**public Item(String category, String name, double price) {**

**this.category = category;**

**this.name = name;**

**this.price = price;**

**}**

**public String getCategory() {**

**return category;**

**}**

**public void setCategory(String category) {**

**this.category = category;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public double getPrice() {**

**return price;**

**}**

**public void setPrice(double price) {**

**this.price = price;**

**}**

**}**

**Item Sorter class**

**import java.util.ArrayList;**

**import java.util.Comparator;**

**import java.util.List;**

**import java.util.stream.Collectors;**

**public class ItemSorter {**

**public static void main(String[] args) {**

**ArrayList<Item> items = new ArrayList<>();**

**items.add(new Item("A", "X", 10.0));**

**items.add(new Item("B", "Y", 11.0));**

**items.add(new Item("C", "Z", 12.0));**

**items.add(new Item("E", "K", 10.0));**

**items.add(new Item("F", "R", 11.0));**

**items.add(new Item("D", "S", 12.0));**

**System.out.println(items);**

**List<Item> naturallySortedList = sortByCategory(items);**

**System.out.println("Sorted with category");**

**naturallySortedList.forEach(x->System.out.println(x.getCategory()));**

**List<Item> explicitlySortedList = sorteByPriceAndName(items);**

**explicitlySortedList.forEach(x->System.out.println(x.getName() + " And "+x.getPrice()));**

**}**

**private static List<Item> sortByCategory(ArrayList<Item> items) {**

**List<Item> sortedList = items.stream()**

**.sorted(Comparator.comparing(Item::getCategory, Comparator.naturalOrder())).collect(Collectors.toList());**

**return sortedList;**

**}**

**private static List<Item> sorteByPriceAndName(ArrayList<Item> items) {**

**List<Item> sortedList = items.stream()**

**.sorted(Comparator.comparing(Item::getPrice).thenComparing(Item::getName)).collect(Collectors.toList());**

**return sortedList;**

**}**

**}**