WEEK 10:

**EX 20.1**

|  |
| --- |
| import java.util.\*; |
|  | import java.io.\*; |
|  |  |
|  | public class Exercise\_20\_01 { |
|  | public static void main(String[] args) throws Exception { |
|  | // Check number of arguments passed |
|  | if (args.length != 1) { |
|  | System.out.println("Usage: java Exercise\_20\_01 TextFile"); |
|  | System.exit(1); |
|  | } |
|  |  |
|  | // Check if file exists |
|  | File textFile = new File(args[0]); |
|  | if (!textFile.exists()) { |
|  | System.out.println("The file " + args[0] + " does not exist."); |
|  | System.exit(2); |
|  | } |
|  |  |
|  | // Create a list of string |
|  | List<String> list = new ArrayList<>(); |
|  |  |
|  | try ( |
|  | // Create input file |
|  | Scanner input = new Scanner(textFile); |
|  | ) { |
|  | while (input.hasNext()) { |
|  | String[] array = input.nextLine().split(" "); |
|  | for (int i = 0; i < array.length; i++) { |
|  | if (array[i].length() > 0 && |
|  | Character.isLetter(array[i].charAt(0))) { |
|  | list.add(array[i]); |
|  | } |
|  | } |
|  | } |
|  | } |
|  | // Sort the list in ascending alphabetical order |
|  | Collections.sort(list); |
|  |  |
|  | // Display the list |
|  | System.out.println(list); |
|  | } |
|  | } |

EX 20.6

|  |
| --- |
| import java.util.\*; |
|  | import java.io.\*; |
|  |  |
|  | public class Exercise\_20\_01 { |
|  | public static void main(String[] args) throws Exception { |
|  | // Check number of arguments passed |
|  | if (args.length != 1) { |
|  | System.out.println("Usage: java Exercise\_20\_01 TextFile"); |
|  | System.exit(1); |
|  | } |
|  |  |
|  | // Check if file exists |
|  | File textFile = new File(args[0]); |
|  | if (!textFile.exists()) { |
|  | System.out.println("The file " + args[0] + " does not exist."); |
|  | System.exit(2); |
|  | } |
|  |  |
|  | // Create a list of string |
|  | List<String> list = new ArrayList<>(); |
|  |  |
|  | try ( |
|  | // Create input file |
|  | Scanner input = new Scanner(textFile); |
|  | ) { |
|  | while (input.hasNext()) { |
|  | String[] array = input.nextLine().split(" "); |
|  | for (int i = 0; i < array.length; i++) { |
|  | if (array[i].length() > 0 && |
|  | Character.isLetter(array[i].charAt(0))) { |
|  | list.add(array[i]); |
|  | } |
|  | } |
|  | } |
|  | } |
|  | // Sort the list in ascending alphabetical order |
|  | Collections.sort(list); |
|  |  |
|  | // Display the list |
|  | System.out.println(list); |
|  | } |
|  | } |

WEEK 11:  
EX21.1

|  |
| --- |
| import java.util.\*; |
|  |  |
|  | public class Exercise\_21\_01 { |
|  | public static void main(String[] args) { |
|  | // Create two linked hash sets |
|  | Set<String> set1 = new LinkedHashSet<>(Arrays.asList( |
|  | "George", "Jim", "John", "Blake", "Kevin", "Michael")); |
|  | Set<String> set2 = new LinkedHashSet<>(Arrays.asList( |
|  | "George", "Katie", "Kevin", "Michelle", "Ryan")); |
|  |  |
|  | // Display the union of the two sets |
|  | Set<String> union = new LinkedHashSet<>(set1); |
|  | union.addAll(set2); |
|  | System.out.println("Union of the two sets: " + union); |
|  |  |
|  | // Display the difference of the two sets |
|  | Set<String> difference = new LinkedHashSet<>(set1); |
|  | difference.removeAll(set2); |
|  | System.out.println("Difference of the two sets: " + difference); |
|  |  |
|  |  |
|  | // Display the intersetion of the two sets |
|  | Set<String> intersection = new LinkedHashSet<>(); |
|  | for (String e: set2) { |
|  | if (set1.contains(e)) |
|  | intersection.add(e); |
|  | } |
|  | System.out.println("Intersection of the two sets: " + intersection); |
|  | } |
|  | } |

EX 21.6

|  |
| --- |
| import java.util.\*; |
|  |  |
|  | public class Exercise\_21\_06 { |
|  | public static void main(String[] args) { |
|  | // Create a Scanner |
|  | Scanner input = new Scanner(System.in); |
|  |  |
|  | // Create a list and two sets |
|  | Map<Integer, Integer> map = new HashMap<>(); // Stores results |
|  |  |
|  | // Prompt the user to enter a number of integers |
|  | System.out.println("Enter a number of integers." |
|  | + "\nInput ends when the input is 0:"); |
|  |  |
|  | int key; |
|  | while ((key = input.nextInt()) != 0) { |
|  | if (!map.containsKey(key)) { |
|  | map.put(key, 1); |
|  | } |
|  | else { |
|  | int frequency = map.get(key); |
|  | frequency++; |
|  | map.put(key, frequency); |
|  | } |
|  | } |
|  |  |
|  | // Find the maximum occurrence |
|  | int max = Collections.max(map.values()); |
|  |  |
|  | // Display the integers that have the most occurences |
|  | System.out.print("The most occurrences integers are: "); |
|  | for (Map.Entry<Integer, Integer> entry : map.entrySet()) { |
|  | if (entry.getValue() == max) { |
|  | System.out.print(entry.getKey() + " "); |
|  | } |
|  | } |
|  | System.out.println(); |
|  | } |
|  | } |

WEEK 12:

EX 24.2

|  |
| --- |
| public class Exercise\_02 { |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | String[] words = new String[10]; |
|  |  |
|  | for (int i = 0; i < words.length; i++) |
|  | words[i] = "Name" + i; |
|  | MyList<String> list = new MyLinkedList<>(words); |
|  |  |
|  | System.out.println(list.contains("Name0")); |
|  | list.set(4, "Arantes Sa"); |
|  | System.out.println(list.indexOf("Luiz")); |
|  | System.out.println(list.lastIndexOf("Name9")); |
|  | System.out.println(list.get(5)); |
|  | System.out.println(list); |
|  | } |
|  |  |
|  | } |

WEEK 13:  
EX 30.4

|  |
| --- |
| import java.util.concurrent.ExecutorService; |
|  | import java.util.concurrent.Executors; |
|  |  |
|  | public class Ex304 { |
|  | private static Integer sum = 0; |
|  | public static void main(String[] args) { |
|  | ExecutorService executor = Executors.newCachedThreadPool(); |
|  |  |
|  | for (int i = 0; i < 1000; i++) { |
|  | executor.execute(new AddOne()); |
|  | } |
|  |  |
|  | executor.shutdown(); |
|  |  |
|  | while (!executor.isTerminated()) { |
|  | } |
|  |  |
|  | System.out.println("sum = " + sum); |
|  | } |
|  |  |
|  | private static class AddOne implements Runnable { |
|  | public void run() { |
|  | sum++; |
|  | } |
|  | } |
|  | } |

EX 30.11

|  |
| --- |
| public class Ex3011 { |
|  |  |
|  | public static void main(String[] args) { |
|  | Object object1 = new Object(); |
|  | Object object2 = new Object(); |
|  | new DeadLockThread(object1, object2, 1); |
|  | new DeadLockThread(object2, object1, 2); |
|  | } |
|  |  |
|  | static class DeadLockThread implements Runnable { |
|  | Object object1; |
|  | Object object2; |
|  | int threadNumber; |
|  |  |
|  | public DeadLockThread(Object object1, Object object2, int threadNumber) { |
|  | this.object1 = object1; |
|  | this.object2 = object2; |
|  | this.threadNumber = threadNumber; |
|  | Thread thread = new Thread(this); |
|  | thread.start(); |
|  | } |
|  |  |
|  | @Override |
|  | public void run() { |
|  | System.out.println("Thread #" + threadNumber + ": starting"); |
|  | while(true) { |
|  | synchronized (object1) { |
|  | System.out.println("Thread #" + threadNumber + ": 1-st synchronized"); |
|  | synchronized (object2) { |
|  | System.out.println("Thread #" + threadNumber + ": 2-nd synchronized"); |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |

WEEK 14:  
EX 31.4

|  |
| --- |
| import java.io.\*; |
|  | import java.net.\*; |
|  | import java.util.Date; |
|  | import javafx.application.Application; |
|  | import javafx.application.Platform; |
|  | import javafx.scene.Scene; |
|  | import javafx.scene.control.ScrollPane; |
|  | import javafx.scene.control.TextArea; |
|  | import javafx.stage.Stage; |
|  |  |
|  | public class MultiThreadServer extends Application { |
|  |  |
|  | private TextArea ta = new TextArea(); |
|  |  |
|  |  |
|  | private int clientNo = 0; |
|  |  |
|  | @Override |
|  | public void start(Stage primaryStage) { |
|  |  |
|  | Scene scene = new Scene(new ScrollPane(ta), 450, 200); |
|  | primaryStage.setTitle("MultiThreadServer"); // Set the stage title |
|  | primaryStage.setScene(scene); // Place the scene in the stage |
|  | primaryStage.show(); // Display the stage |
|  |  |
|  | new Thread( () -> { |
|  | try { |
|  |  |
|  | ServerSocket serverSocket = new ServerSocket(8000); |
|  | ta.appendText("MultiThreadServer started at " |
|  | + new Date() + '\n'); |
|  |  |
|  | while (true) { |
|  |  |
|  | Socket socket = serverSocket.accept(); |
|  |  |
|  |  |
|  | clientNo++; |
|  |  |
|  | Platform.runLater( () -> { |
|  |  |
|  | ta.appendText("Starting thread for client " + clientNo + |
|  | " at " + new Date() + '\n'); |
|  |  |
|  |  |
|  | InetAddress inetAddress = socket.getInetAddress(); |
|  | ta.appendText("Client " + clientNo + "'s host name is " |
|  | + inetAddress.getHostName() + "\n"); |
|  | ta.appendText("Client " + clientNo + "'s IP Address is " |
|  | + inetAddress.getHostAddress() + "\n"); |
|  | }); |
|  |  |
|  |  |
|  | new Thread(new HandleAClient(socket)).start(); |
|  | } |
|  | } |
|  | catch(IOException ex) { |
|  | System.err.println(ex); |
|  | } |
|  | }).start(); |
|  | } |
|  |  |
|  |  |
|  | class HandleAClient implements Runnable { |
|  | private Socket socket; |
|  | public HandleAClient(Socket socket) { |
|  | this.socket = socket; |
|  | } |
|  |  |
|  |  |
|  | public void run() { |
|  | try { |
|  |  |
|  | DataInputStream inputFromClient = new DataInputStream( |
|  | socket.getInputStream()); |
|  | DataOutputStream outputToClient = new DataOutputStream( |
|  | socket.getOutputStream()); |
|  |  |
|  |  |
|  | while (true) { |
|  | // Receive radius from the client |
|  | double radius = inputFromClient.readDouble(); |
|  |  |
|  |  |
|  | double area = radius \* radius \* Math.PI; |
|  |  |
|  |  |
|  | outputToClient.writeDouble(area); |
|  |  |
|  | Platform.runLater(() -> { |
|  | ta.appendText("radius received from client: " + |
|  | radius + '\n'); |
|  | ta.appendText("Area found: " + area + '\n'); |
|  | }); |
|  | } |
|  | } |
|  | catch(IOException e) { |
|  | ex.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  | } |