A blue sign with white text

Description automatically generated

**UNIVERSITY OF BOLTON**

**BSc COMPUTING**

**COURSEWORK SUBMISSION FORM**

|  |
| --- |
| **Student/Centre to complete:**    SURNAME/FAMILY NAME: **SABIR** FORENAMES: **NAVEED**    BOLTON STUDENT ID: **2224755** EMAIL: **ns7crt@bolton.ac.uk**    DATE OF SUBMISSION: **08/12/2023**    MODULE NO./TITLE: SWE5202 Data Structures and Algorithms    TUTOR’S NAME:………Abdul Razak…………………...... ……………………    COURSEWORK TITLE: Portfolio item 3 Tree Data structures and recursion    Please state if this is your FIRST submission OR REFERRED/DEFERRED submission OR a REPEAT submission?  **FIRST**    …………………..…………………………………………………………………………………………  **Declaration**  **I hereby declare that this work is my own work. I understand that if I am suspected of plagiarism or another form of cheating, my work be referred to Academic Registrar and/or the Board of Examiners, which may result in me being expelled from the programme. I understand once I submit this work, it will automatically belong to the University of Bolton.** |

|  |
| --- |
| **Creative Technologies** |
| **Course / Programme: BEng (Hons) in Software Engineering**  **Module name and code: Data Structures and Algorithms SWE5202**  **Tutor: Abdul Razak**  **Assessment Number: 3**  **Assessment Title: Tree Data structures and recursion**  **Weighting: 25%**  **Issue Date: W/C 20 November 2023**  **Submission Deadline: 08 December** **2023 @16:00.** |

[Figure 1 4](#_Toc152902963)

[Figure 2 4](#_Toc152902964)

[Figure 3 4](#_Toc152902965)

[Figure 4 5](#_Toc152902966)

[Figure 5 5](#_Toc152902967)

[Figure 6 6](#_Toc152902968)

[Figure 7 6](#_Toc152902969)

[Figure 8 6](#_Toc152902970)

[Figure 9 7](#_Toc152902971)

[Figure 10 8](#_Toc152902972)

[Figure 11 8](#_Toc152902973)

[Figure 12 9](#_Toc152902974)

[Figure 13 9](#_Toc152902975)

[Figure 14 9](#_Toc152902976)

[Figure 15 9](#_Toc152902977)

[Figure 16 10](#_Toc152902978)

[Figure 17 10](#_Toc152902979)

[Figure 18 10](#_Toc152902980)

[Figure 19 10](#_Toc152902981)

[Figure 20 11](#_Toc152902982)

[Figure 21 11](#_Toc152902983)

[Figure 22 11](#_Toc152902984)

[Figure 23 12](#_Toc152902985)

[Figure 24 13](#_Toc152902986)

[Figure 25 13](#_Toc152902987)

[Figure 26 13](#_Toc152902988)

[Figure 27 13](#_Toc152902989)

[Figure 28 13](#_Toc152902990)

[Figure 29 14](#_Toc152902991)

[Figure 30 14](#_Toc152902992)

[Figure 31 14](#_Toc152902993)

[Figure 32 15](#_Toc152902994)

[Figure 33 15](#_Toc152902995)

[Figure 34 16](#_Toc152902996)

Contents

[Tree Data structures and recursion 3](#_Toc152905750)

[STAGE 1 4](#_Toc152905751)

[STAGE 2 6](#_Toc152905752)

[STAGE 3 8](#_Toc152905753)

[STAGE 4 10](#_Toc152905754)

[STAGE 5 13](#_Toc152905755)

[UML 16](#_Toc152905756)

[REFERENCES 17](#_Toc152905757)

# Tree Data structures and recursion

In computer science and data management, the effort of designing and implementing advanced tree data structures and traversal algorithms is crucial (Thomas H. Cormen et al., 2022). Trees, a basic data structure, are widely utilised in many fields, including database management systems, file systems, and network routing algorithms (Introduction to the Analysis of Algorithms by Robert Sedgewick and Philippe Flajolet. ). The challenge is to create complex tree structures that efficiently arrange and store data while allowing for quick retrieval and change. (J. Á. Velázquez-Iturbide, M. E. Castellanos and R. Hijón-Neira, 2016) Developing optimised traversal algorithms that allow for seamless transit throughout the tree's hierarchical branches is also critical. These efforts significantly increase the efficiency and speed of algorithms, which has a positive influence on the overall performance of software systems and applications that rely on structured data representation. The study and improvement of such complicated tree structures and traversal algorithms will remain critical to the evolution of computing paradigms. (H. P. Zellweger, 2016)

# STAGE 1

This code's objective is to define the Employee Java class, which represents an employee and includes basic information such forename, surname, job title, and remuneration. The class has two constructors: one for creating workers with default settings and another for creating employees with specified data (Algorithms, 4th Edition by Robert Sedgewick and Kevin Wayne. ).

A screenshot of a computer program

Description automatically generated

Figure 1

Setter methods are available for changing certain attributes, and getter methods are available for getting the values of these components.

A black background with green and blue text

Description automatically generated

Figure 2

A black background with blue green and white text

Description automatically generated

Figure 3

The class implements the Cloneable interface to ease object cloning, allowing the creation of a clone of an existing Employee object. As a result, the clone technique is disabled (Data Structures & Algorithms in Java. ).

A black background with colorful text

Description automatically generated

Figure 4

Furthermore, the toString function has been changed to provide a string representation of the Employee object for debugging and logging.

A computer screen shot of a program code

Description automatically generated

Figure 5

# STAGE 2

The goal of the code is to show how to create and display an employee hierarchy in Java Swing using a tree structure provided with DefaultMutableTreeNode (Introduction to Tree - Data Structure and Algorithm Tutorials.2021).



Figure 6

Employee information, such as forename, surname, job title, and remuneration, is stored in the Employee class. The DisplayTest001 class produces employee instances, builds a tree structure by connecting them with DefaultMutableTreeNode objects, and then shows the hierarchy with a recursive method (displayTree).

A screen shot of a computer program

Description automatically generated

Figure 7

The main function generates tree nodes for each employee, builds the hierarchy by adding child nodes, and then reports the hierarchical structure using the displayTree method.

A screen shot of a computer code

Description automatically generated

Figure 8

The output visually illustrates the organization's staff hierarchy, emphasising the connections between distinct job positions and their related personnel. The code demonstrates how to use tree structures for hierarchical data representation, which is useful in contexts such as organisational charts and file system topologies.

A black screen with many small colored text

Description automatically generated with medium confidence

Figure 9

# STAGE 3

The goal of the code is to show how to generate and present an employee hierarchy using a tree structure created in Java Swing using DefaultMutableTreeNode. In addition to producing the hierarchy, the DisplayTest002 class has methods for filtering and presenting information about the personnel structure. It specifically covers two techniques: displayHighEarners and displayWithSubordinates (Joshua Bloch, 2018).

The main method generates employee instances, constructs a hierarchical structure with tree nodes, and then demonstrates the capabilities of the other methods.

A screen shot of a computer

Description automatically generated

Figure 10

The first approach, displayHighEarners, recursively examines the employee hierarchy and shows information about those earning more than a given threshold (in this case, £50,000).

A screen shot of a computer code

Description automatically generated

Figure 11

The second approach, displayWithSubordinates, recursively discovers workers with subordinates, publishes their information, and shows the number of direct reports, including subordinate information.

A screen shot of a computer program

Description automatically generated

Figure 12

A screen shot of a computer code

Description automatically generated

Figure 13

The code is a practical illustration of using tree structures to manage and visualise hierarchical data, exhibiting its adaptability in extracting specific information from the hierarchy. It is especially effective in cases where identifying employees with subordinates or filtering employees based on compensation is crucial, such as in organisational management or reporting systems (How to Use Trees (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components). ).

A screen shot of a computer

Description automatically generated

Figure 14

A screen shot of a computer screen

Description automatically generated

Figure 15

# STAGE 4

The purpose of the code in **DisplayTest003** is to create and display a graphical representation of an employee hierarchy using Java Swing components, specifically **JTree** and **JFrame**. Here's an overview of the key functionality (Trail: Creating a GUI With Swing (The Java™ Tutorials)):

1. **Tree Model and JTree Creation:**
   * The code creates a **DefaultTreeModel** named **treeModel** with the root node representing the top-level employee in the hierarchy.

A close up of a text

Description automatically generated

Figure 16

* + A **JTree** named **tree** is instantiated using the created **DefaultTreeModel**. The **JTree** is a Swing component designed for displaying hierarchical data in tree structures.

A black background with white text

Description automatically generated

Figure 17

1. **JFrame Setup:**
   * A **JFrame** named **frame** is created with the title "Employee Hierarchy."
   * The JFrame is configured to exit the application when the close button is clicked (**JFrame.EXIT\_ON\_CLOSE**).

A screen shot of a computer screen

Description automatically generated

Figure 18

1. **Adding JTree to JFrame:**
   * The **JTree** is embedded in a **JScrollPane** to allow scrolling if the hierarchy exceeds the visible area.
   * The **JScrollPane** is added to the content pane of the **JFrame**.

A black background with text

Description automatically generated

Figure 19

1. **Employee Data and Tree Structure:**
   * Employee instances are created, each representing a node in the hierarchy. These instances include information such as forename, surname, job title, and salary.

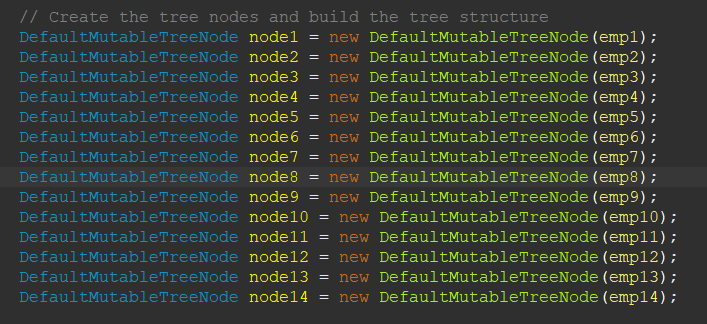


Figure 20

* + **DefaultMutableTreeNode** objects are created for each employee, and the hierarchy is built by adding child nodes to represent reporting relationships within the organization.

A screen shot of a computer program

Description automatically generated

Figure 21

1. **Display:**
   * The **JTree** is displayed within the **JFrame**, providing a visual representation of the employee hierarchy.
   * The **JFrame** is set to a size of 400x600 pixels and is positioned at the centre of the screen.

A screen shot of a computer code

Description automatically generated

Figure 22

* + The **JFrame** is set to be visible, making the employee hierarchy represented by the **JTree** visible to the user.

A screenshot of a computer

Description automatically generated

Figure 23

Overall, the code demonstrates how to use Java Swing components to produce a rudimentary graphical representation of hierarchical data. In this example, the approach is used to depict an organisational people hierarchy, but it may be used to other situations where tree topologies are useful for data visualisation. (J. P. Vieira Costa et al., 2022)

# STAGE 5

The provided code is a Java program that demonstrates the creation and manipulation of an employee hierarchy using a tree structure implemented with **DefaultMutableTreeNode** and **JTree**. Here's a brief explanation of the code:

1. **Class Overview:**
   * **DisplayTest004**: This class serves as the main class for the application.
2. **Main Method (public static void main(String[] args)):**
   * Creates a **DefaultTreeModel** with the root node representing the initial employee hierarchy.

A close up of a text

Description automatically generated

Figure 24

* + Creates a **JTree** using the tree model.

A black background with white text

Description automatically generated

Figure 25

* + Adds new nodes to the tree model using the **addNewNodes** method.

A black background with white text

Description automatically generated

Figure 26

* + Updates the hierarchy based on the specified criteria using the **updateHierarchy** method.

A black background with white text

Description automatically generated

Figure 27

* + Displays the employee hierarchy in a **JFrame** using a **JTree**.

A screen shot of a computer screen

Description automatically generated

Figure 28

1. **Create Employee Tree Method (private static DefaultMutableTreeNode createEmployeeTree())):**
   * Creates a tree structure representing an initial employee hierarchy with various employees and their roles.

A screenshot of a computer screen

Description automatically generated

Figure 29

1. **Add New Nodes Method (private static void addNewNodes(DefaultTreeModel treeModel)):**
   * Adds new employee nodes to the existing tree model.

A screen shot of a computer code

Description automatically generated

Figure 30

1. **Update Hierarchy Method (private static void updateHierarchy(DefaultTreeModel treeModel)):**
   * Updates the hierarchy as specified in the code. In this case, it relocates the subordinates of "Node 7" to report to "Node 5" and removes "Node 7."

A screen shot of a computer program

Description automatically generated

Figure 31

1. **Find Node Method (private static DefaultMutableTreeNode findNode(DefaultMutableTreeNode root, String forename, String surname)):**
   * Finds a node with a specific forename and surname in the employee hierarchy tree.
   * Calls the recursive method **findNodeRecursively** to perform the search.

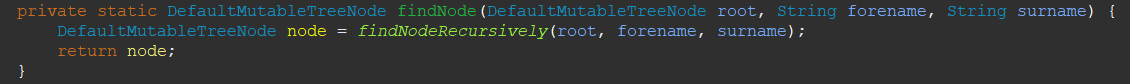


Figure 32

1. **Find Node Recursively Method (private static DefaultMutableTreeNode findNodeRecursively(DefaultMutableTreeNode currentNode, String forename, String surname)):**
   * Recursively searches for a node with a specific forename and surname in the employee hierarchy tree.
   * Returns the node if found; otherwise, returns **null**.

A computer screen with colorful text

Description automatically generated

Figure 33

The goal of this code is to demonstrate how to construct, manipulate, and display an employee hierarchy in a Java Swing application using a tree structure. It shows how to add new nodes and update the hierarchy, demonstrating the versatility of the DefaultMutableTreeNode and JTree classes in maintaining hierarchical data. (A. Al-Rawi, A. Lansari and F. Bouslama, 2003)

# UML

A group of text boxes

Description automatically generated

Figure 34

# REFERENCES

# H. P. Zellweger (2016) Tree Visualizations in Structured Data Recursively Defined by the Aleph Data Relation. In:*2016 20th International Conference Information Visualisation (IV)* . , [Online] Available from: . [Accessed].

# J. Á. Velázquez-Iturbide, M. E. Castellanos and R. Hijón-Neira (2016) Recursion Removal as an Instructional Method to Enhance the Understanding of Recursion Tracing. *IEEE Transactions on Education,* [Online] 59 (3), pp. 161-168 Available from: . [Accessed].

# A. Al-Rawi, A. Lansari and F. Bouslama (2003) A new non-recursive algorithm for binary search tree traversal. In:*10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003* . , [Online] Available from: . [Accessed].

J. P. Vieira Costa, C. Atreio Portela, H. Kimura, M. Ladeira and F. Barros Diniz (2022) Lifetime Probability of Default using Survival Tree-Based Models. In:*2022 17th Iberian Conference on Information Systems and Technologies (CISTI)* . , [Online] Available from: . [Accessed].

Trail: Creating a GUI With Swing (The Java™ Tutorials). ( ) *,* [Online] ( ), pp. Available from: [https://docs.oracle.com/javase/tutorial/uiswing/.](https://docs.oracle.com/javase/tutorial/uiswing/) [Accessed].

How to Use Trees (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components). ( ) *,* [Online] ( ), pp. Available from: <https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html.> [Accessed].

Data Structures & Algorithms in Java. ( ) *,* [Online] ( ), pp. Available from: <https://everythingcomputerscience.com/books/schoolboek-data_structures_and_algorithms_in_java.pdf.> [Accessed].

Introduction to the Analysis of Algorithms by Robert Sedgewick and Philippe Flajolet. ( ) *,* [Online] ( ), pp. Available from: [https://aofa.cs.princeton.edu/home/.](https://aofa.cs.princeton.edu/home/) [Accessed].

Algorithms, 4th Edition by Robert Sedgewick and Kevin Wayne. ( ) *,* [Online] ( ), pp. Available from: [https://algs4.cs.princeton.edu/home/.](https://algs4.cs.princeton.edu/home/) [Accessed].

Joshua Bloch (2018) *Effective Java  
Third Edition.*Third Edition ed. [Online] : . Available from: <https://kea.nu/files/textbooks/new/Effective%20Java%20%282017%2C%20Addison-Wesley%29.pdf.> [Accessed].

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2022) *Introduction to Algorithms  
Fourth Edition.*ed. [Online] : . Available from: <https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf.> [Accessed].

*Introduction to Tree - Data Structure and Algorithm Tutorials.* (2021) ed. [Online] : .Available from: [https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/.](https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/) [Accessed].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In this assessment I have achieved the following objectives**. | | | |  |
| **Tick appropriate box**  ***NA – not attempted : Part – part completed : Full - fully completed*** | | **NA** | **Part** | **Full** |
| Stage 1 – | Employee class created with shallow copying |  |  | x |
| Stage 2 – | Used recursion to display employees |  |  | x |
| Stage 3 – | Used recursion to display employees earning over a certain salary  Used recursion to display supervisors and the number of employees working directly for them |  |  | x |
| Stage 4 – | Display the model as a JTree |  |  | x |
| Stage 5 – | Modify the tree contents as required |  |  | x |