HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Opbject Oriented Programming

TOPIC: VISUALIZATION OF OPERATIONS ON TREE DATA STRUCTURES

ONE LOVE. ONE FUTURE.

Task Assignment

	Nguyễn Song Hào	Nguyễn Nam Hải	Hà Hoàng Hiệp	Trần Thị Hiền
Package exception (6 classes)	NoneAlgorithmSpecifiedException TreeException TreeNotBalancedException	NodeExistedException NodeFullChildrenException NodeNotExistsException		
Package operation (7 classes)	DeleteMakeBalancedPressed	CreatePressed DeletePressed InsertPressed SearchPressed UpdatePressed UserAction		
Package controller	GenericTreeController (traverse operation) BalancedTreeController (delete operation)	GenericTreeController (the other 5 operations and 2 buttons: Reset & Undo) BalancedTreeController (insert operation) BinaryTreeController (100%) BalancedTreeController (100%) BalancedBinaryTreeController (100%)	GenericTreeController (2 buttons: Back & Help) HelpController (100%)	
Package fxml		MainWindow.fxml (100%) GenericTree.fxml (100%)	Help.fxml (100%)	
Package test (1 class)		MainWindowApplication (100%)		
Package treedatastructure	GenericTree (100%) BalancedTree (100%) Node (all except for animation)	BinaryTree (100%) BalancedBinaryTree (100%) Node (animation in addChild method)	Node (animation in constructor)	
Use Case diagram	60%		40%	
General Class diagram	50%	50%		
Slide	30%	30%		40%
Report	15%	15%	10%	60%

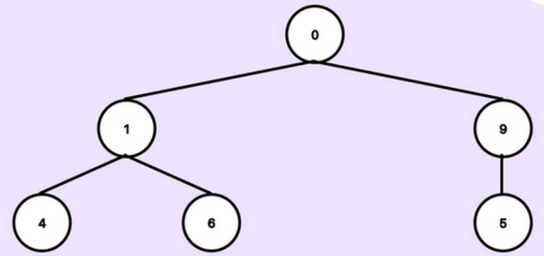
Agenda

- Introduction
- Use Case Diagram
- Project Design
  - General Class Diagram
  - Detail Class Diagram
- Explanation of OOP techniques design
- Demo

Introduction

The project aims to:

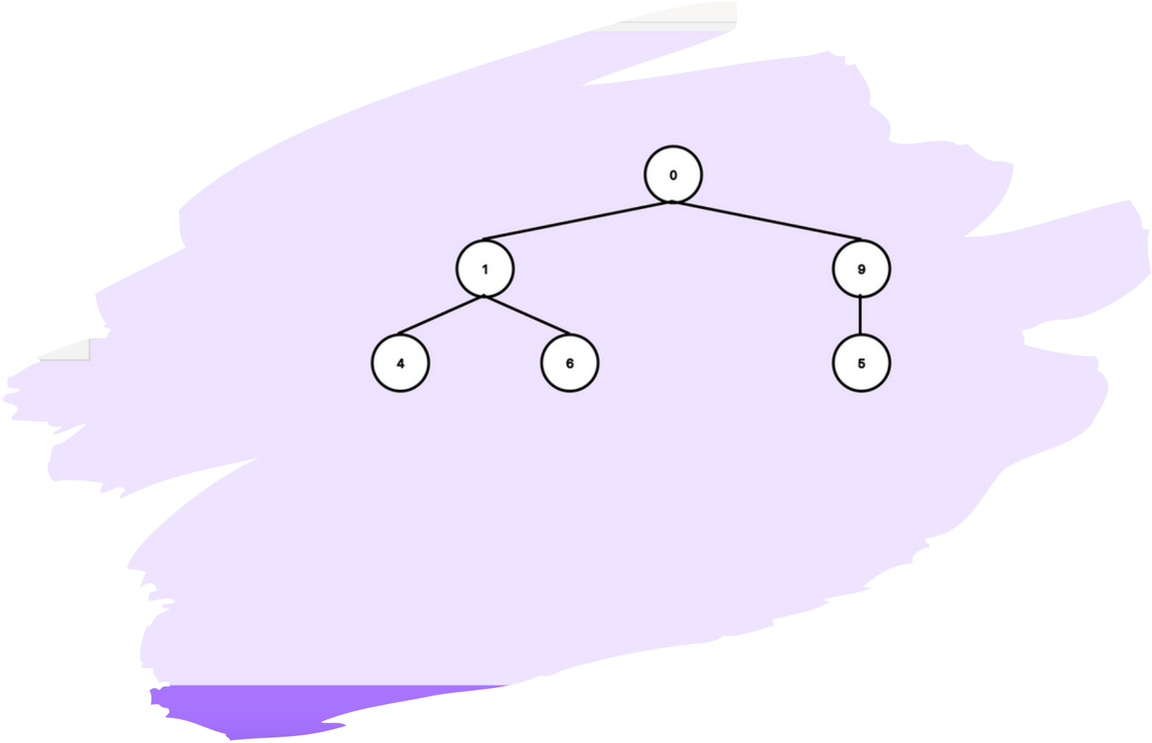
- design a program that allows visualize and understand basic operations on different types of trees
- the development of a user-friendly interface, clearly presenting and explaining basic tree operations, and ensuring interactivity and usability.



Introduction

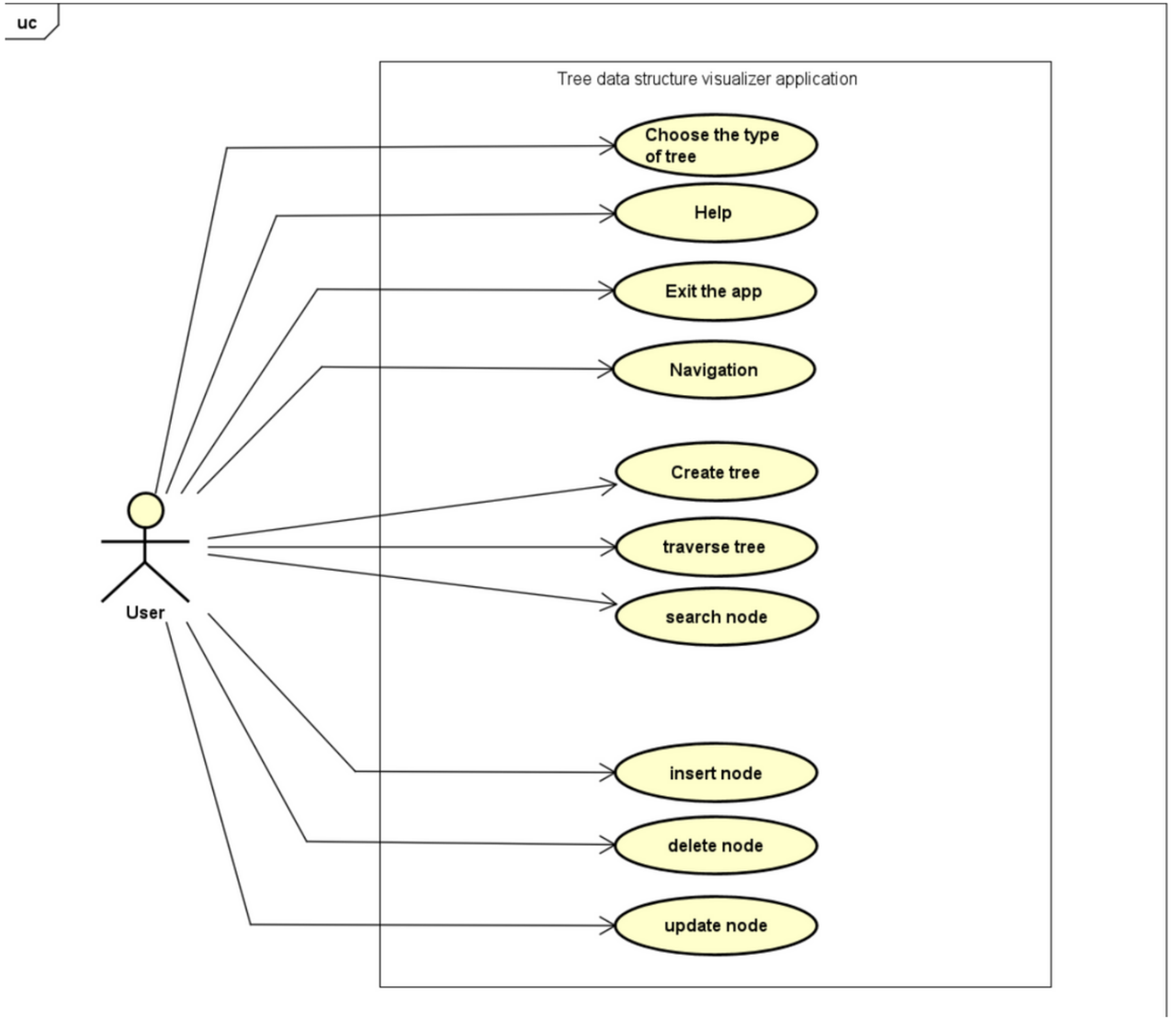
The project includes:

- **4 types of trees**: a generic tree, a binary tree, a balanced tree, and a balanced binary tree
- **6 operations**: creating a new tree, inserting, deleting, updating nodes, searching for nodes, and traversing the tree



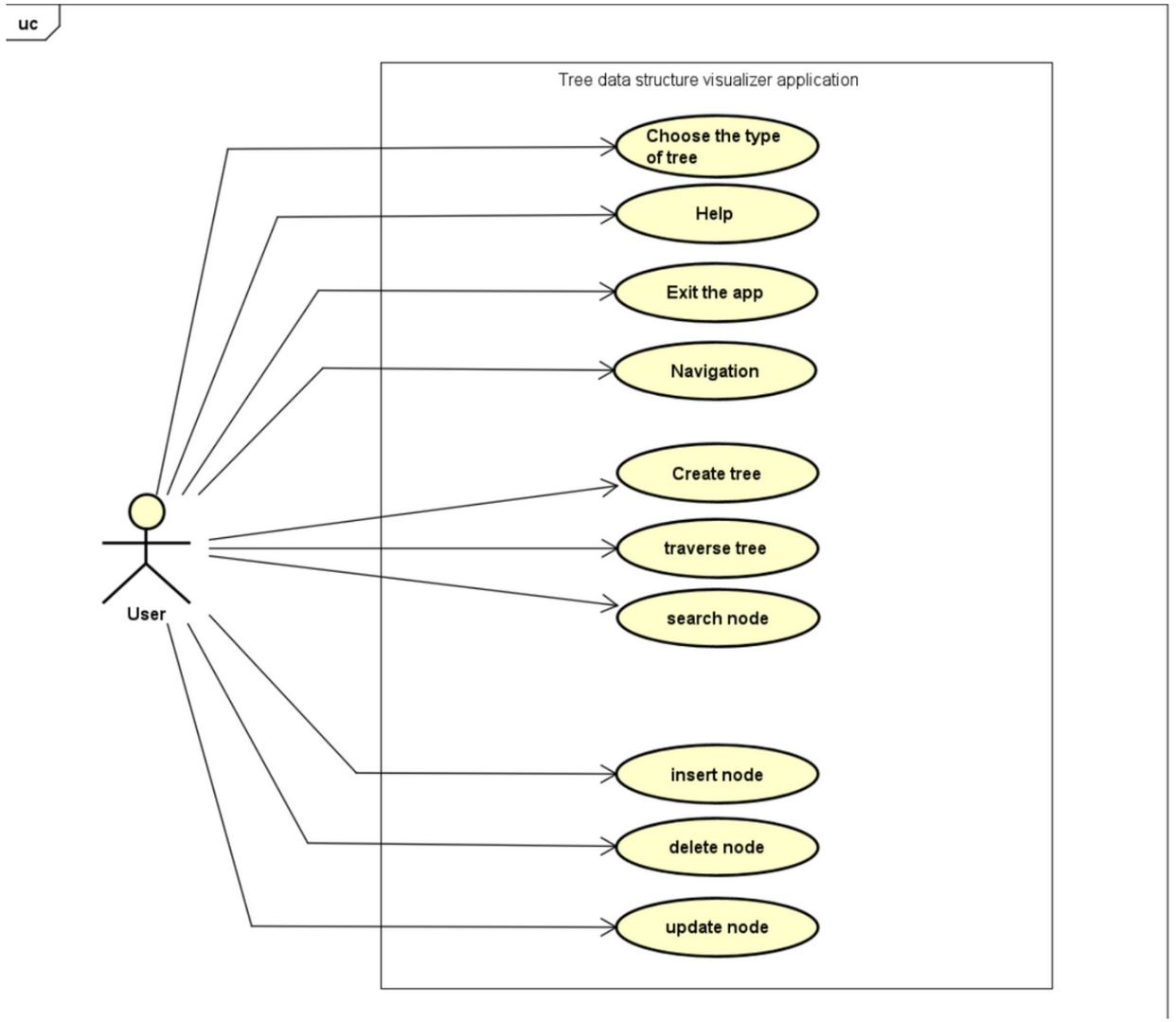
2. Use Case Diagram

- user-friendly interface for choosing tree types and visualizing
- user can perform essential tree operations



2. Use Case Diagram

- "Help" button offers insights into the tree type being visualized
- "Exit" button allows for easy program termination



3. Project Design

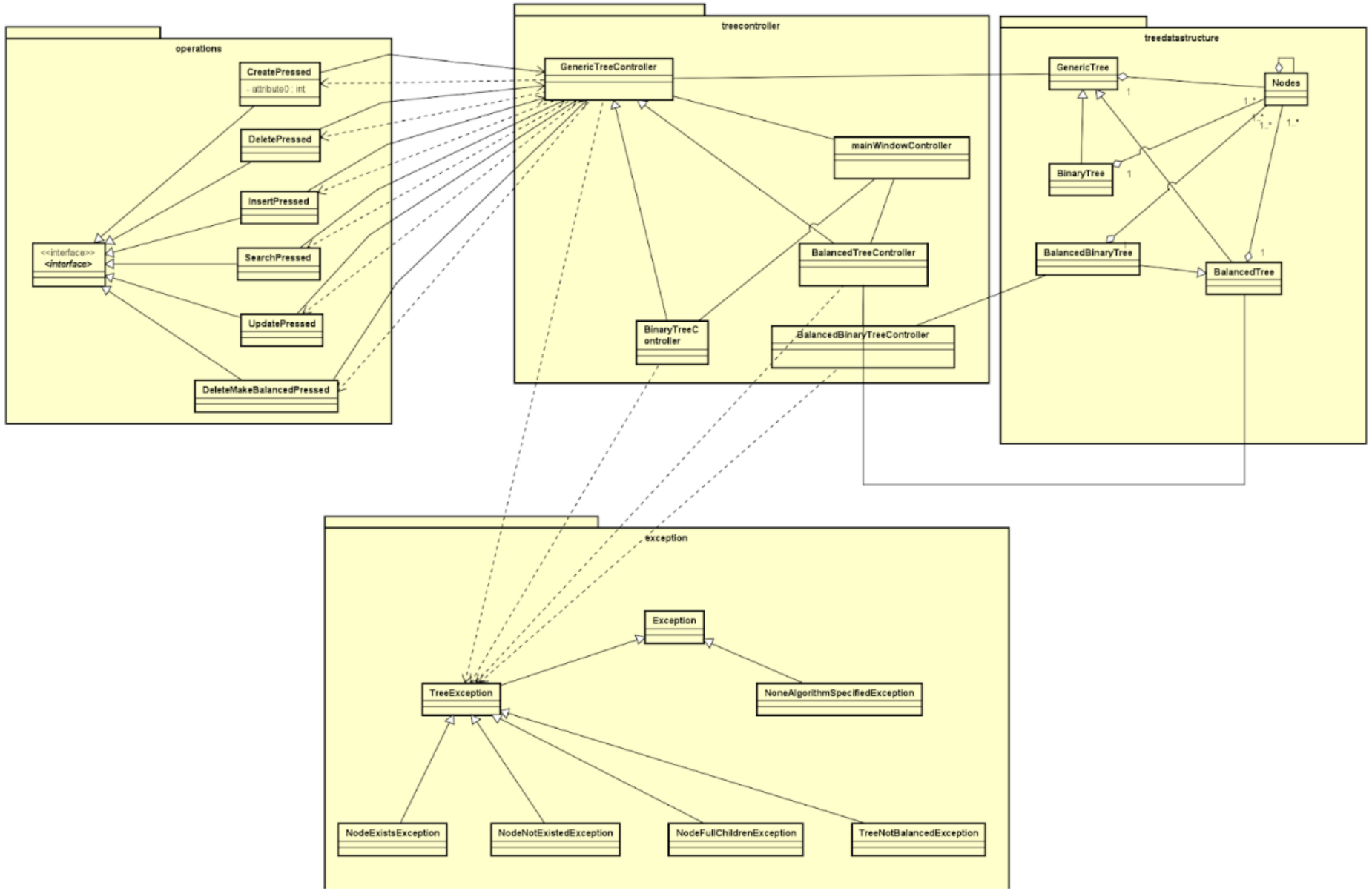
3.1. General Class Diagram

3.2. Detail Class Diagram

3.1. General Class Diagram

Four main packages:

- Operation packages
- Tree data structure package
- Tree controller package
- Exception package



3.2. Detail Class Diagram

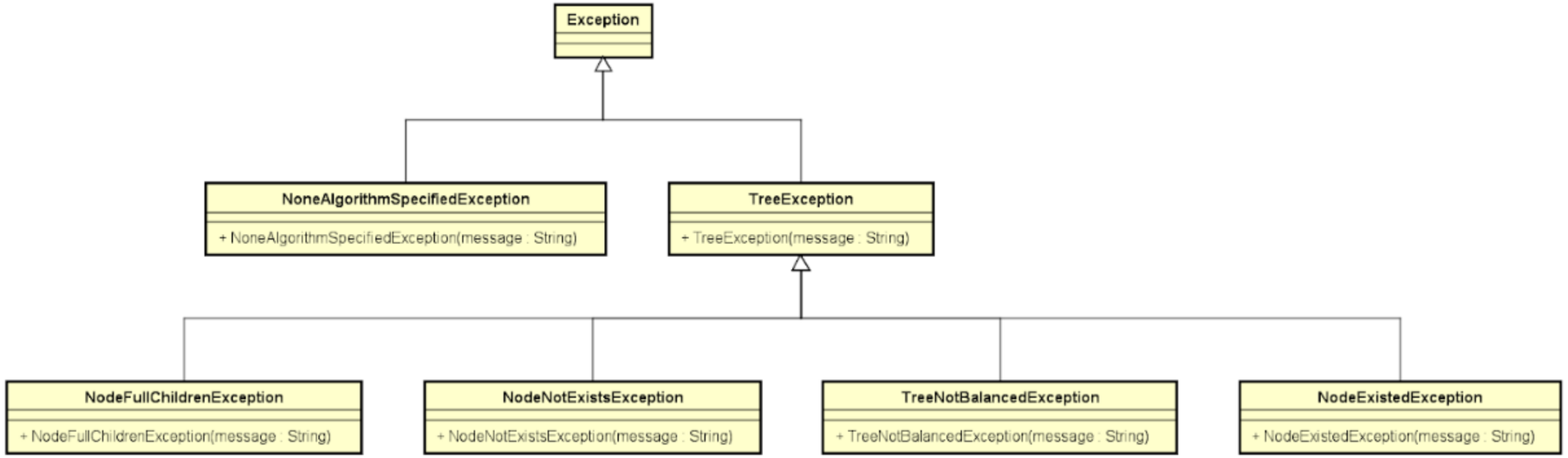
3.2.1. Exception Class Diagram

3.2.2. Data Structure Class Diagram

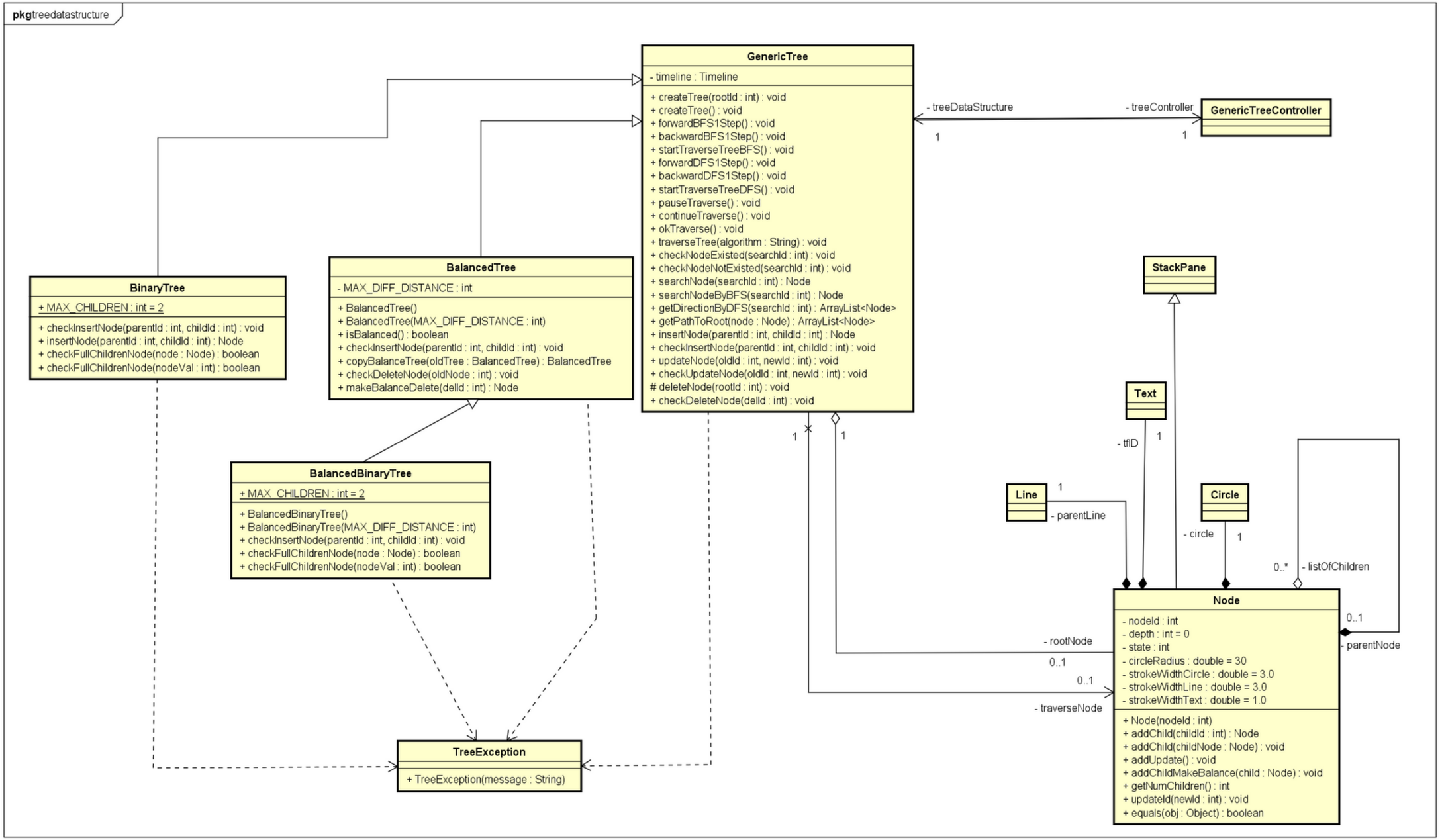
3.2.3. Operation Class Diagram

3.2.4. Controller Class Diagram

3.2.1. Exception Class Diagram



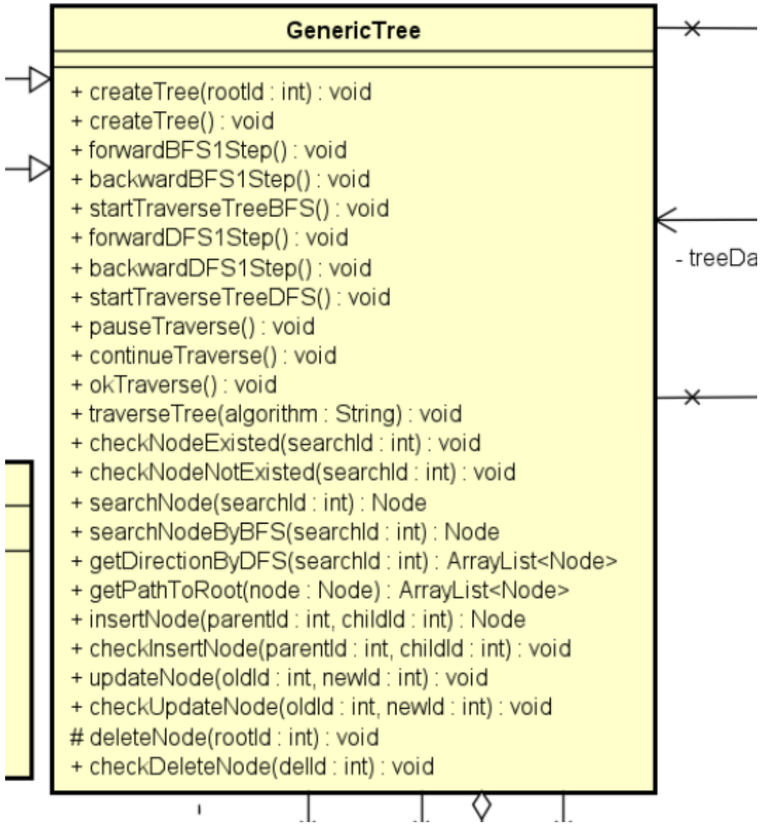
3.2.2. Tree Data Structure Class Diagram



3.2.2. Tree Data Structure Class Diagram

Generic Tree class

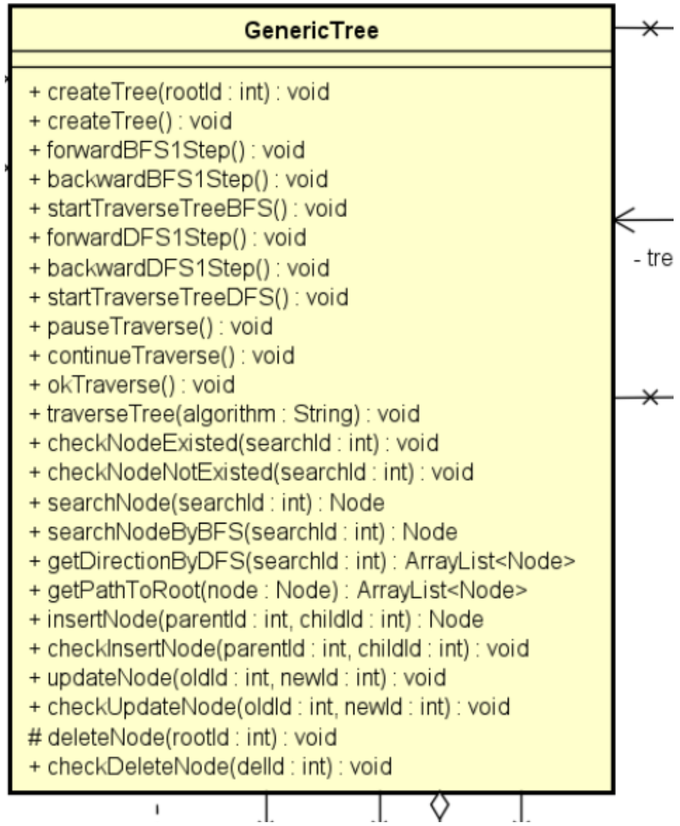
- A generic tree is a tree structure where each node can have an arbitrary number of children nodes.
- Six operations are supported in the Generic Tree Class, including insert, delete, search, traverse, create, update.



3.2.2. Tree Data Structure Class Diagram

Generic Tree class

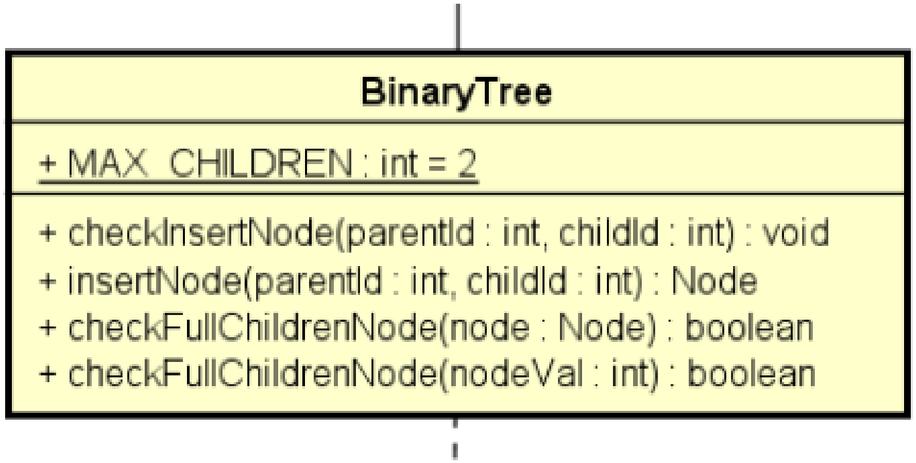
- For each operation, corresponding exceptions are handled, ensuring robustness in the tree's manipulation.
- The class also implements checking algorithms to validate the operations, ensuring the tree remains consistent.



3.2.2. Tree Data Structure Class Diagram

Binary Tree class

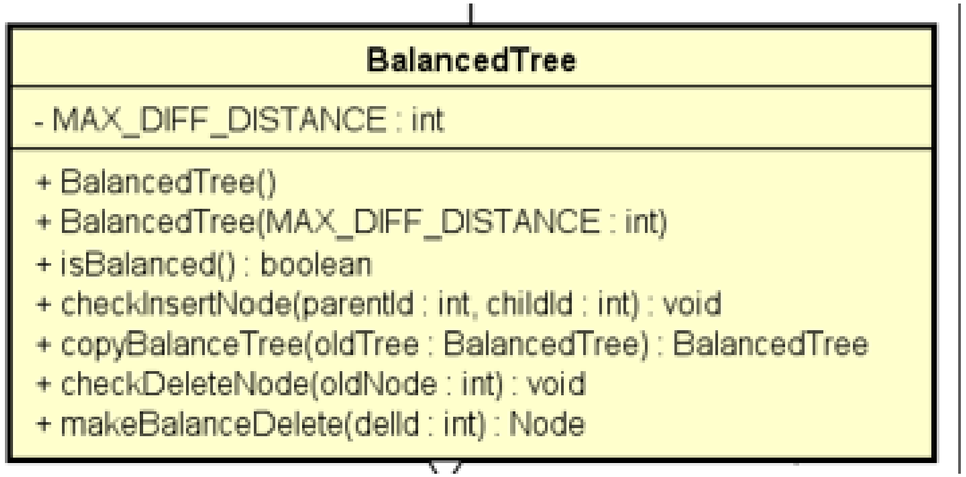
- The Binary Tree Class is a subclass of the Generic Tree Class with modifications to handle binary trees.
- The insert operation is enhanced to check if a node can accept more children or raise NodeFullChildrenException.
- Other operations are inherited from the Generic Tree Class, ensuring a unified interface for tree manipulation.



3.2.2. Tree Data Structure Class Diagram

Balanced Tree Class

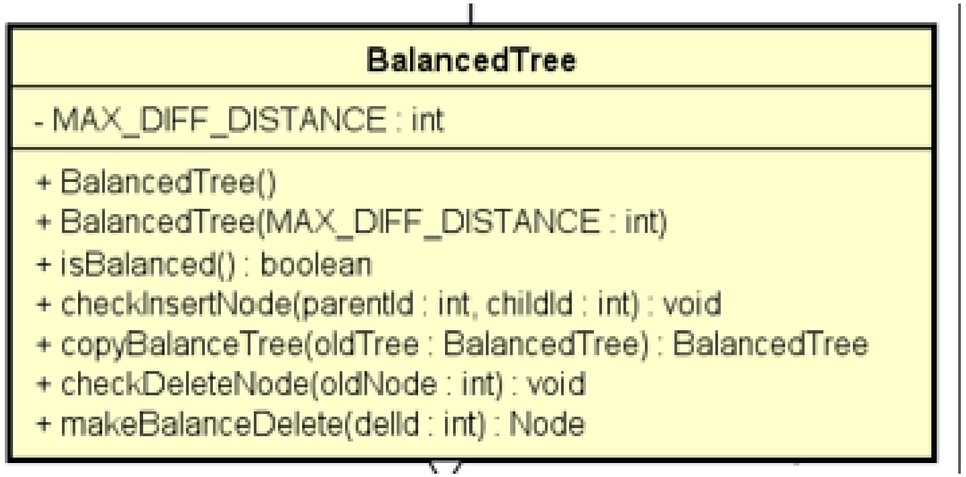
- The Balanced Tree Class represents a balanced tree data structure that maintains depth difference among leaf nodes, ensuring efficiency.
- It inherits from the Generic Tree Class and adds two algorithms for insert and delete operations to maintain balance.



3.2.2. Tree Data Structure Class Diagram

Balanced Tree Class

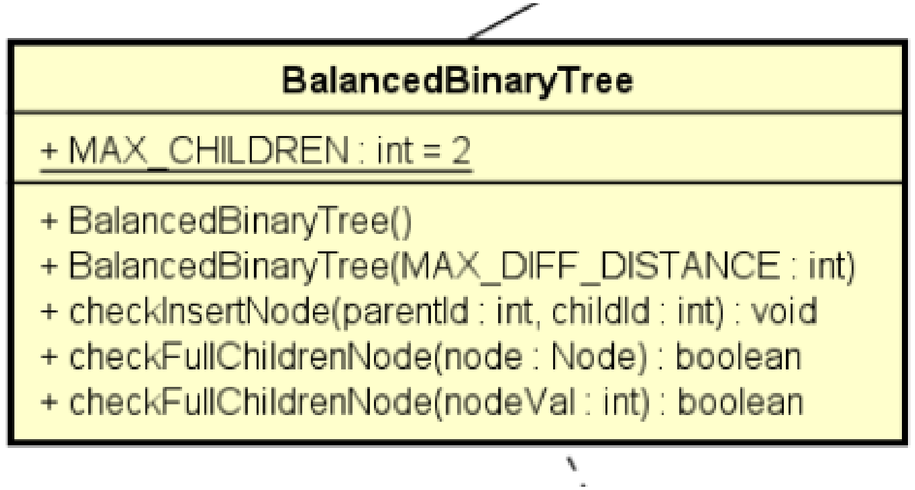
- If an invalid operation makes the tree unbalanced, the user can choose to undo or proceed with the operation.
- When proceeding, our algorithms re-balance the tree, which may involve changing some nodes' parent nodes.



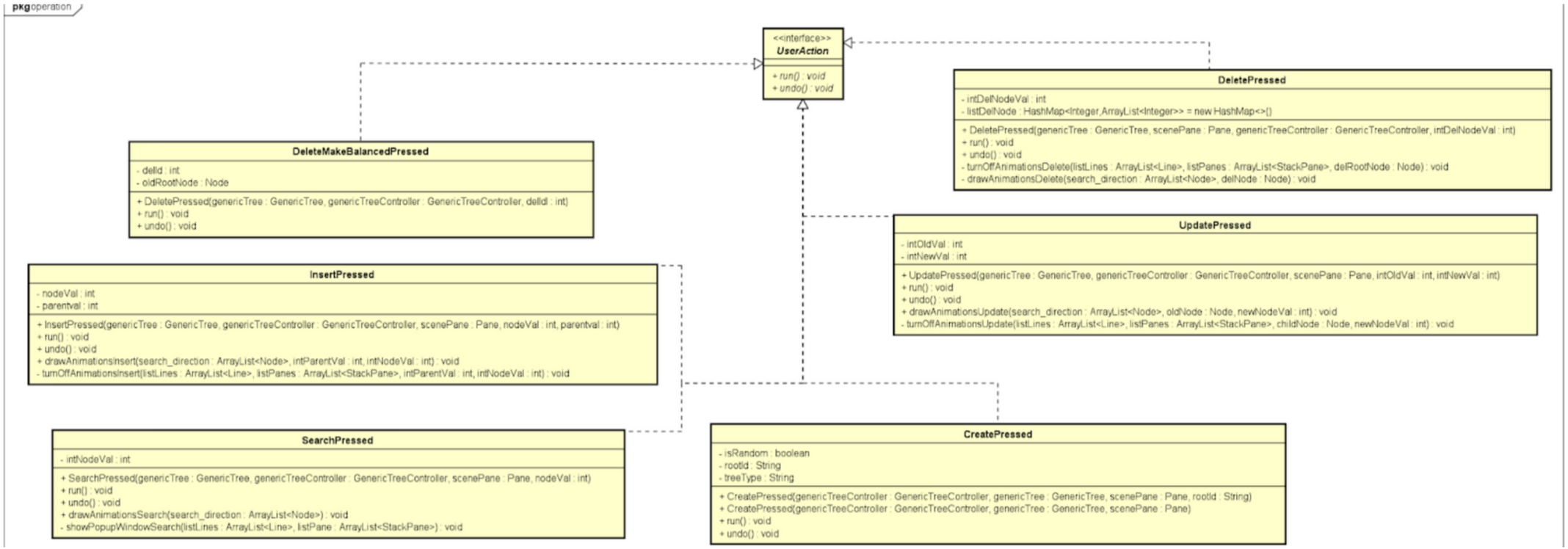
3.2.2. Tree Data Structure Class Diagram

Balanced Binary Tree Class

- The Balanced Binary Tree Class is a subclass of the Balanced Tree Class, tailored for binary trees.
- It introduces additional methods and logic to maintain the binary property during insertions.
- The class ensures that the binary tree remains balanced and efficient while supporting all generic tree and balanced tree operations.



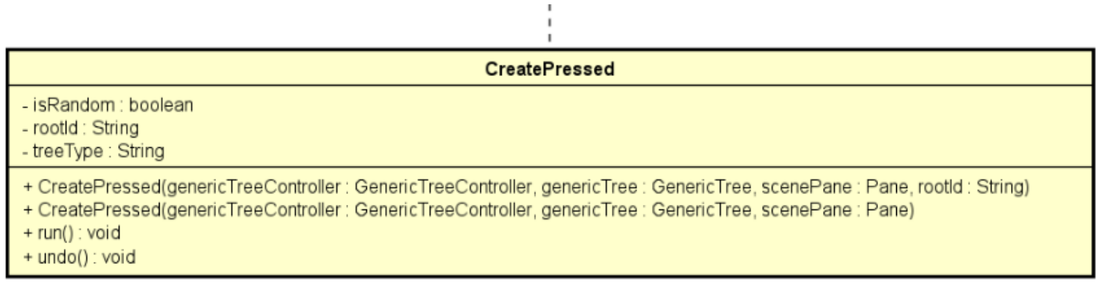
3.2.3. Operation Class Diagram



3.2.3. Operation Class Diagram

CreatePressed Class:

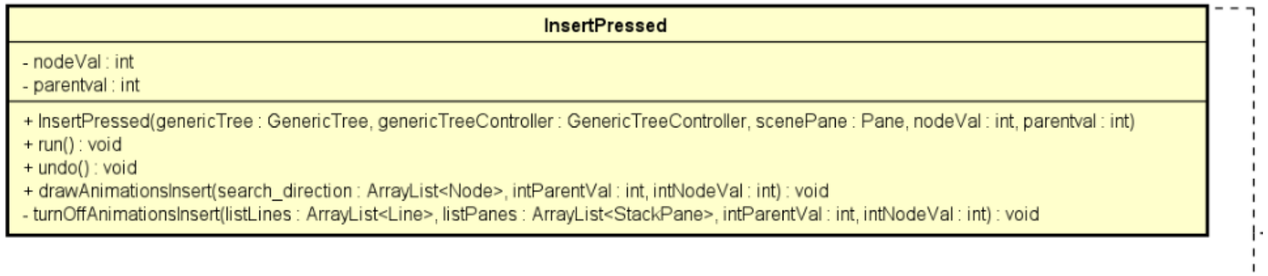
- Attributes: isRandom (boolean), rootID (String)
- run(): Creates a tree based on user's choice, either random or manual input.
- undo(): Clears the tree by setting its root to null and removes it from the scenePane.



3.2.3. Operation Class Diagram

InsertPressed Class:

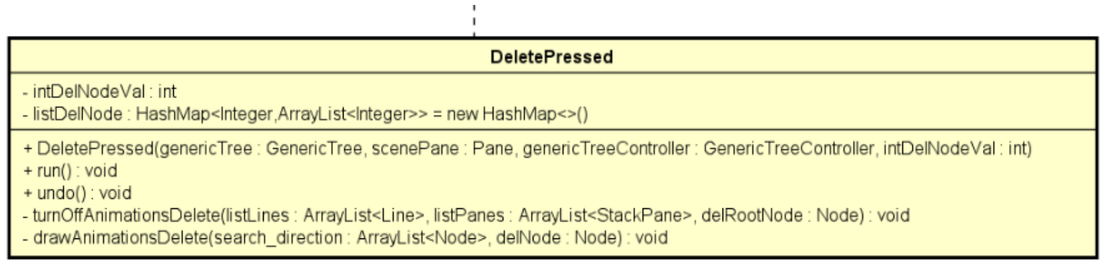
- Attributes: parenVal (int), nodeVal (int), parent (Node)
- run(): Inserts a node into the tree, highlighting the path from root to parent node.
- undo(): Removes the child node from the tree and GUI to revert the insertion.



3.2.3. Operation Class Diagram

DeletePressed Class:

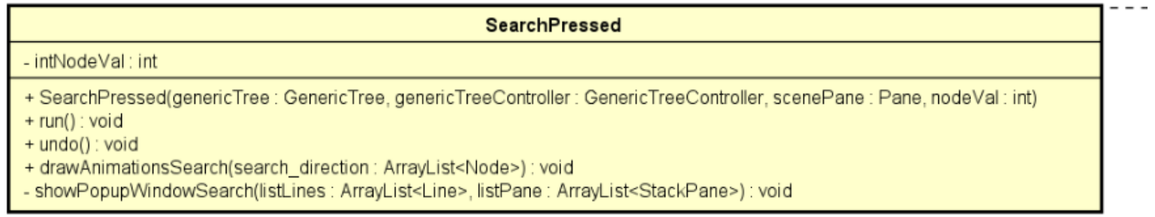
- Attributes: intDelNodeVal (int), parentDelNode (Node), listDelNode (HashMap)
- run(): Deletes a node from the tree, saving subtree information for undo.
- undo(): Restores the deleted node and its subtree to the parent node.



3.2.3. Operation Class Diagram

SearchPressed Class:

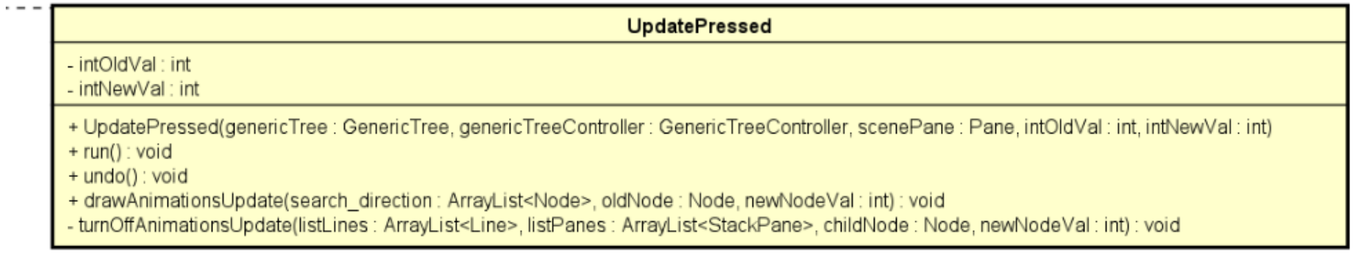
- Attribute: intNodeVal (int)
- run(): Searches for a node, highlighting the path from root to the search node.
- undo(): No action required for undo since static operation



3.2.3. Operation Class Diagram

UpdatePressed Class:

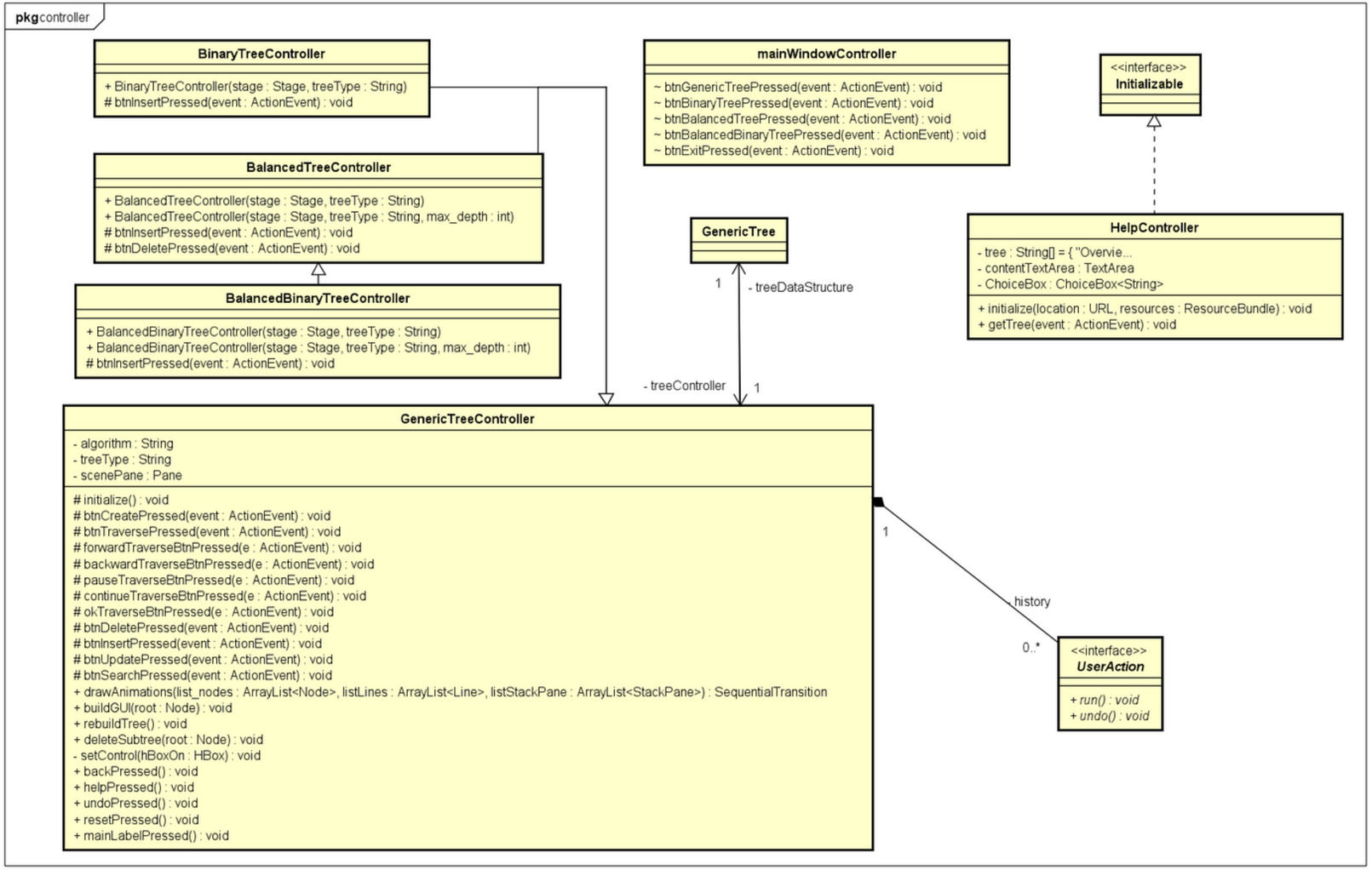
- Attributes: inOldVal (int), intNewVal (int)
- run(): Updates a node's ID, highlighting the path from root to the node.
- undo(): Reverts the node's ID to the old value.



3.2.4. Controller Class Diagram

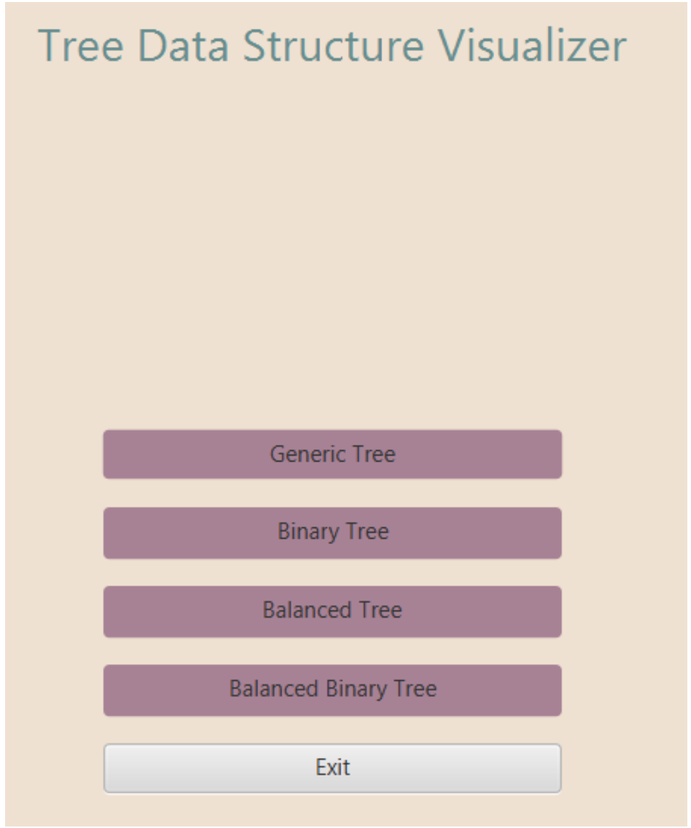
The package Controller contains 6 controllers:

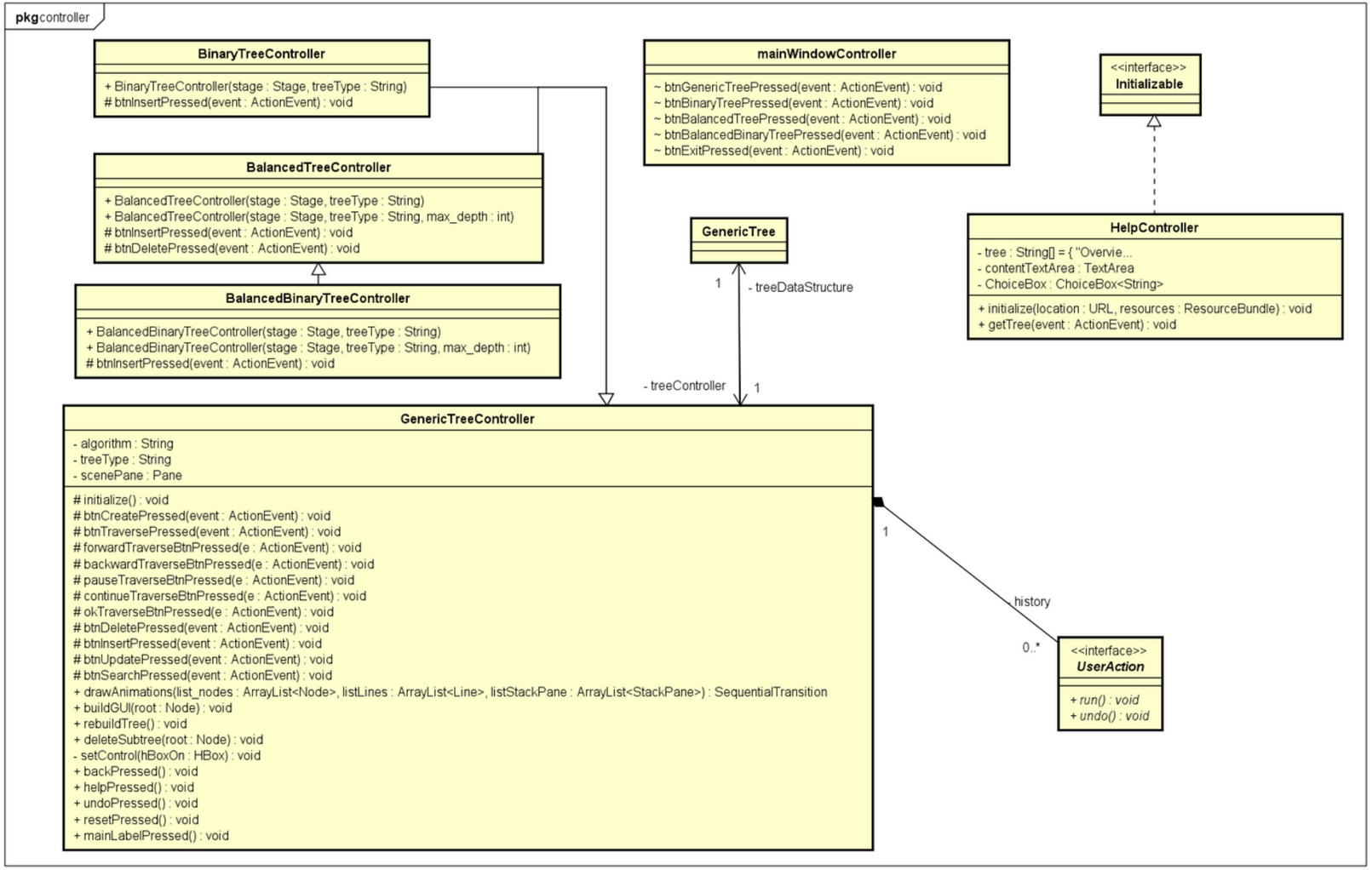
- MainWindowController
- HelpController
- GenericTreeController
- BalancedTreeController
- BinaryTreeController
- BalancedBinaryTreeController



3.2.4. Controller Class Diagram

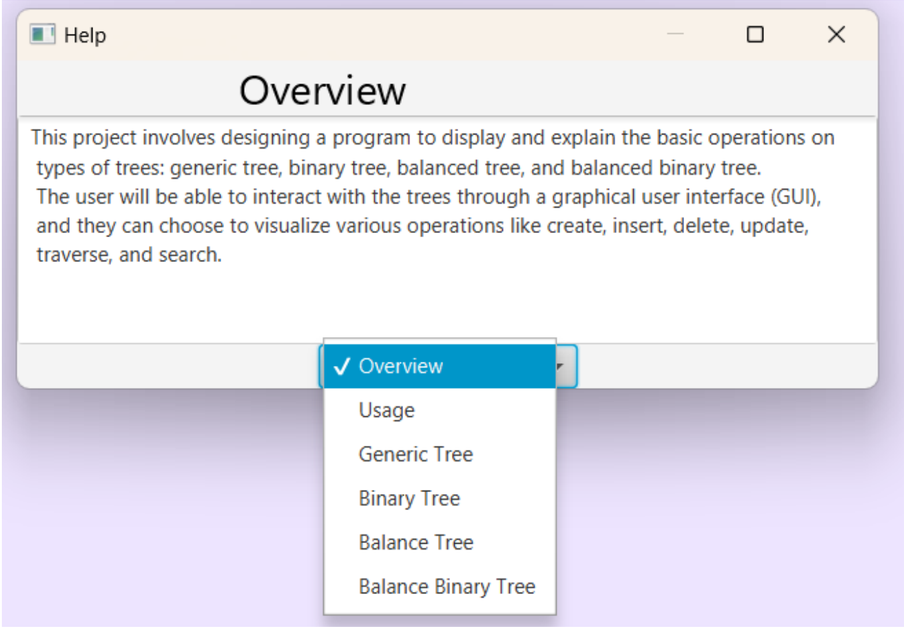
MainWindowController

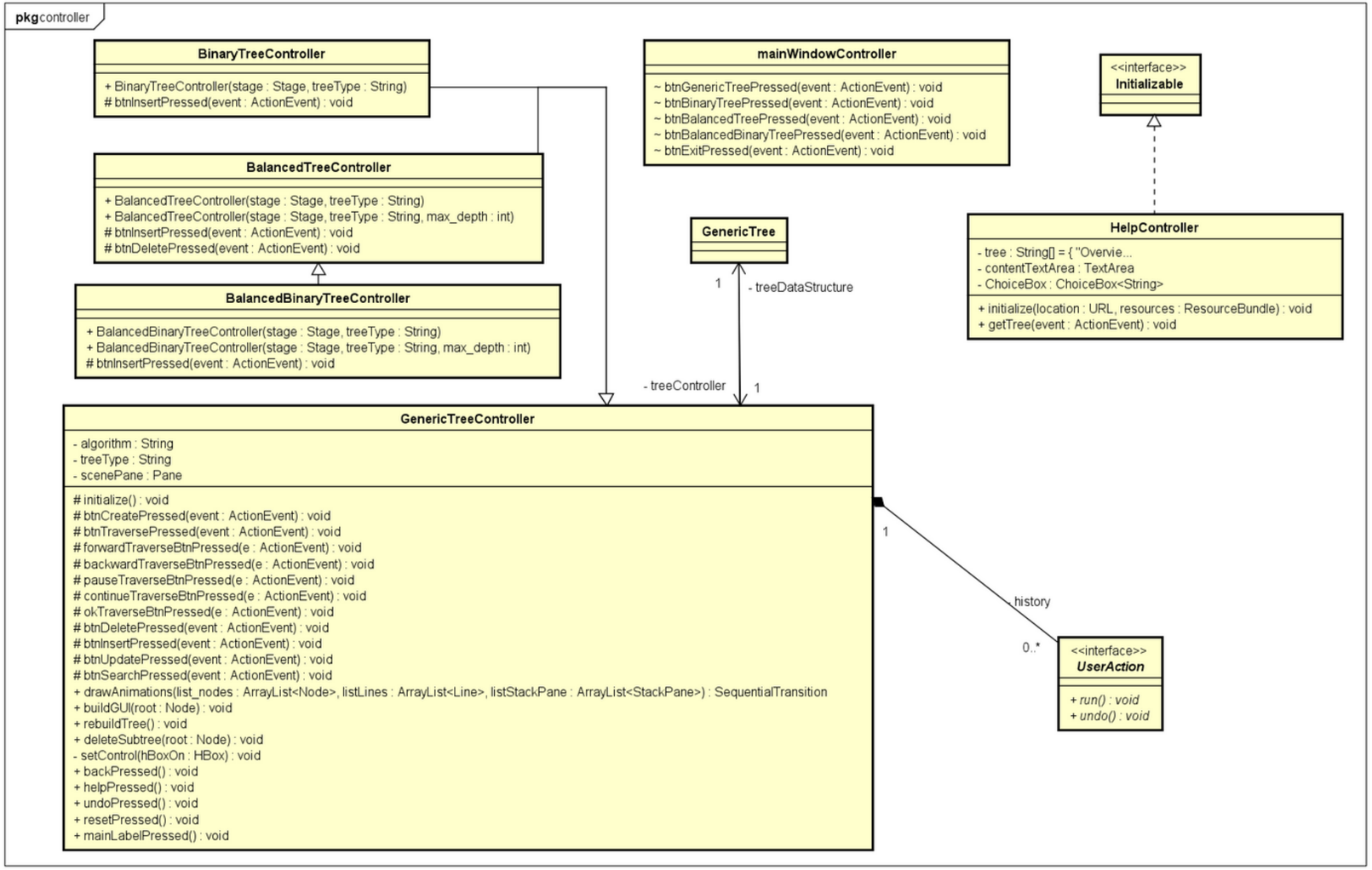




3.2.4. Controller Class Diagram

HelpController





3.2.4. Controller Class Diagram

GenericController

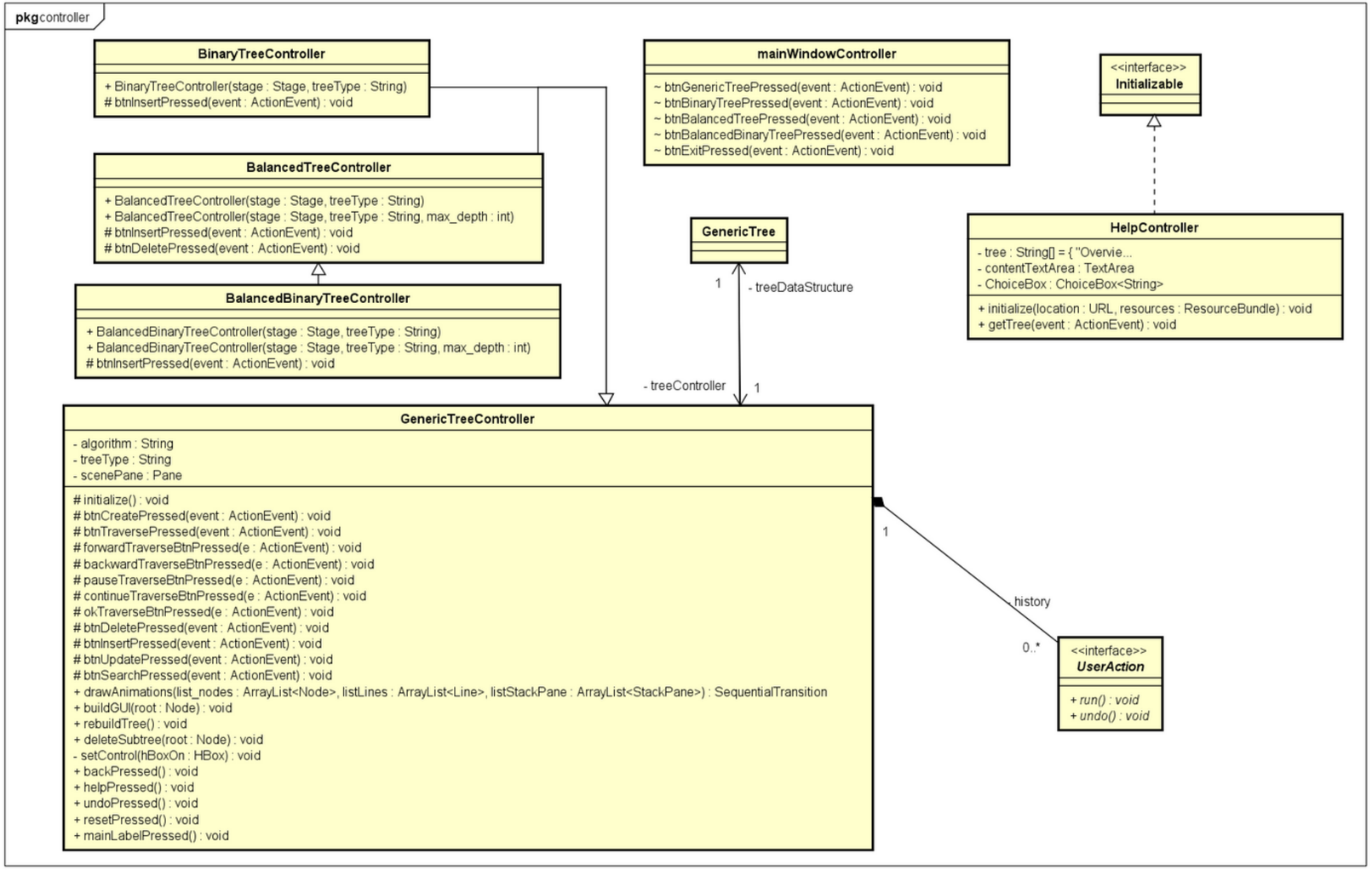
Has 3 main attributes.

Controls 6 basic operations:

- Create
- Search
- Traverse
- Insert
- Delete
- Update

  and 4 other ones:

- Back, Undo
- Reset, Help

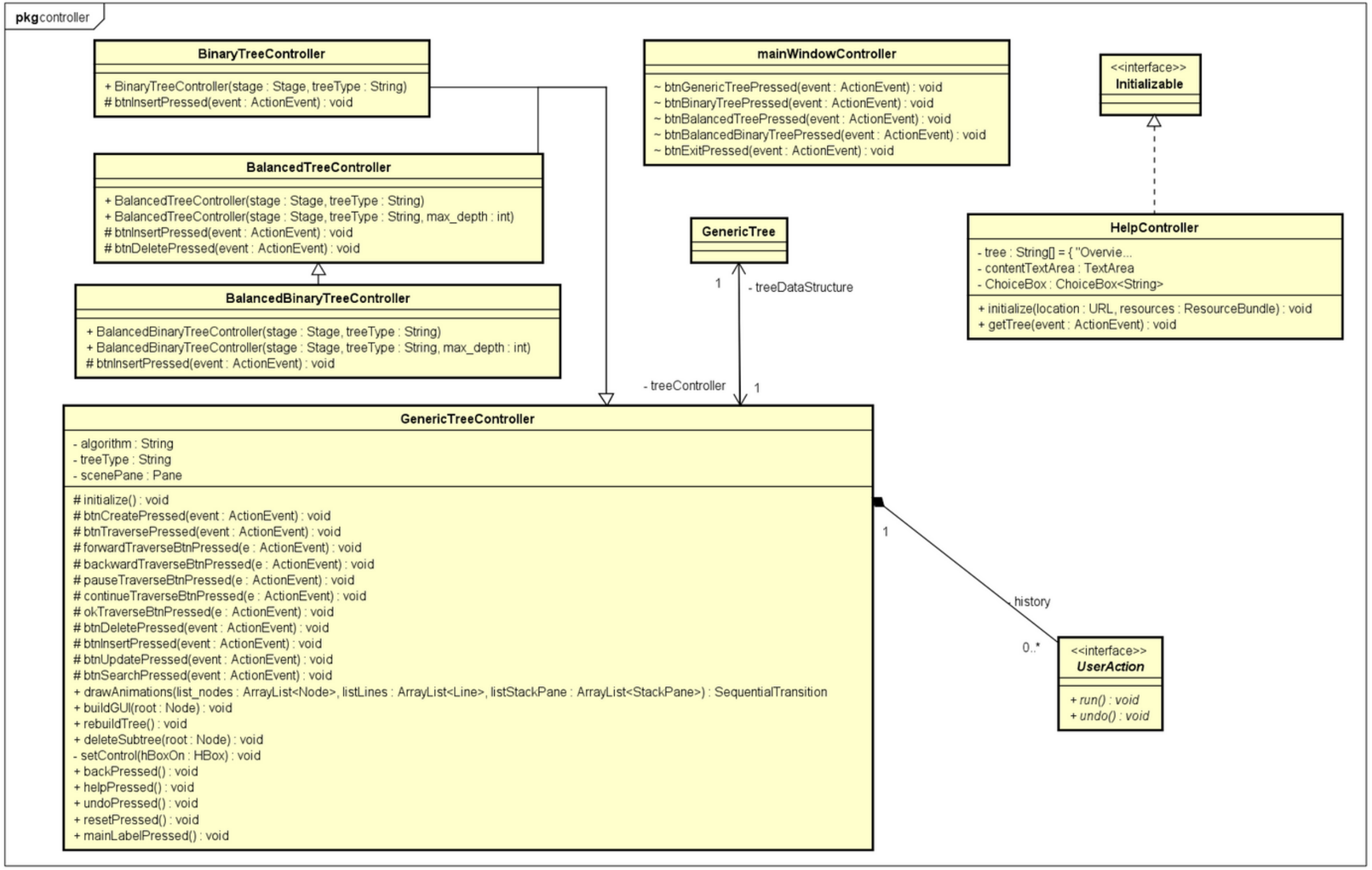


3.2.4. Controller Class Diagram

GenericTreeController

Except the Traverse, handle functions in the form:

- Get the ID
- Check the conditions
- Create a Pressed object.
- run()
- Add the object to **history**



3.2.4. Controller Class Diagram

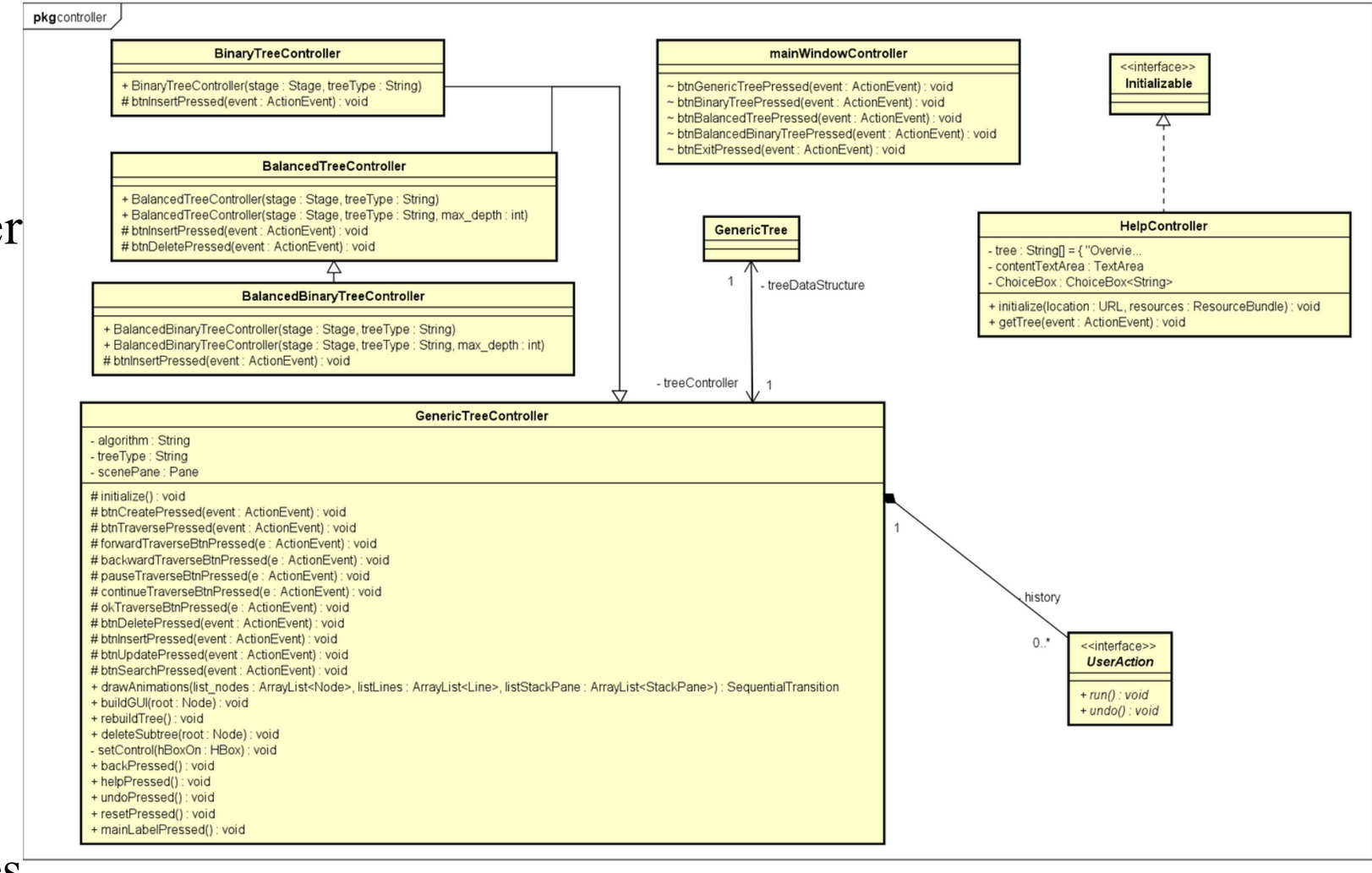
BalancedTreeController & BinaryTreeController

Inherit from GenericTreeController

- Create
- Traverse
- Search
- Update

  BalancedTreeController overrides
- Insert
- Delete

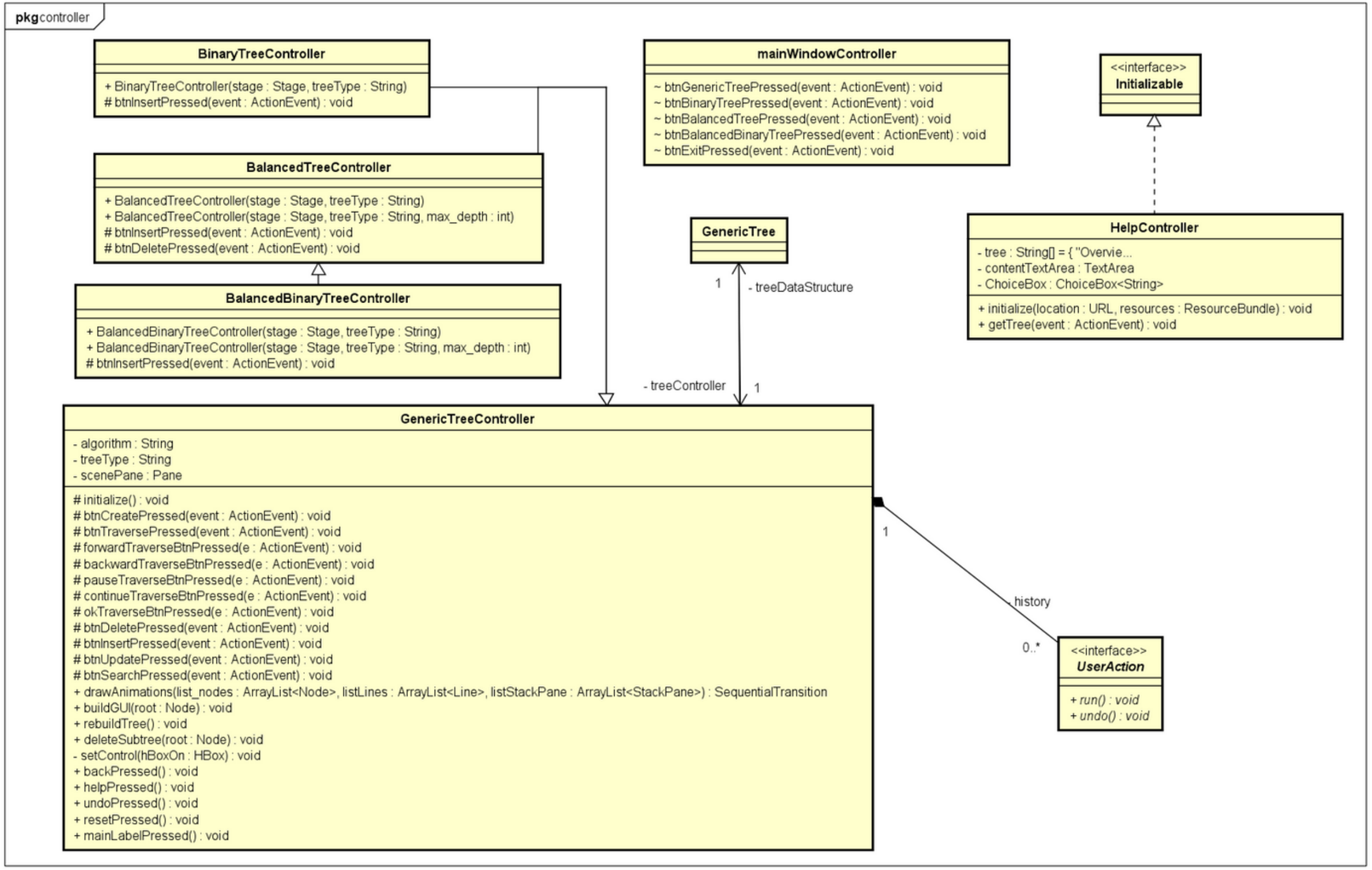
  BinaryTreeController overrides
- Insert



3.2.4. Controller Class Diagram

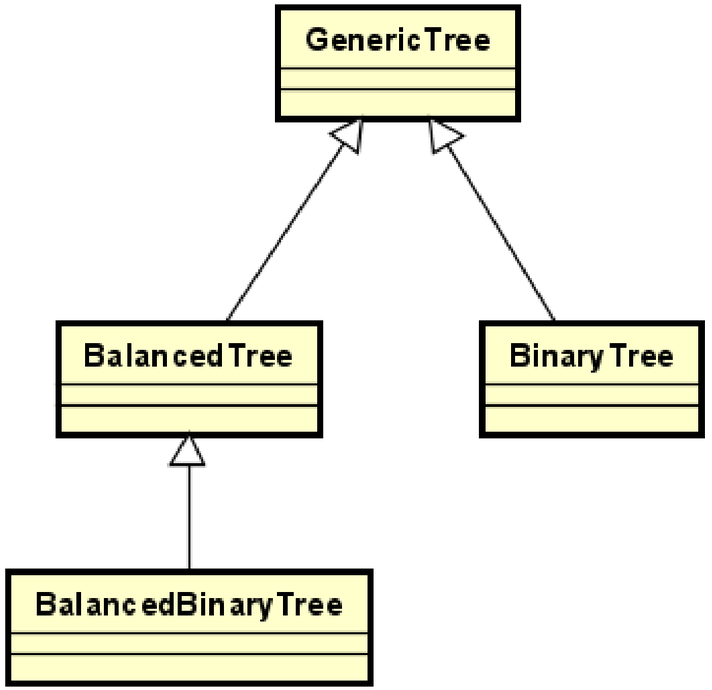
BalancedBinaryTreeController

- Inherits from BalancedTreeController.
- Overrides the method Insert

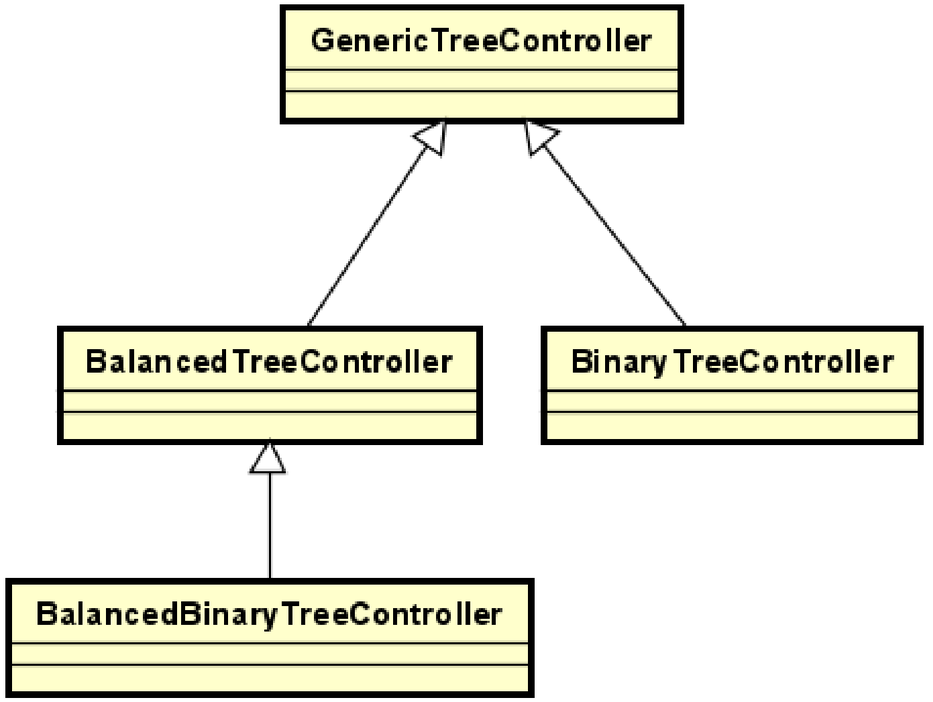


4. Explaination of OOP techniques

Inheritance



package treedatastructure



package controller

4. Explaination of OOP techniques

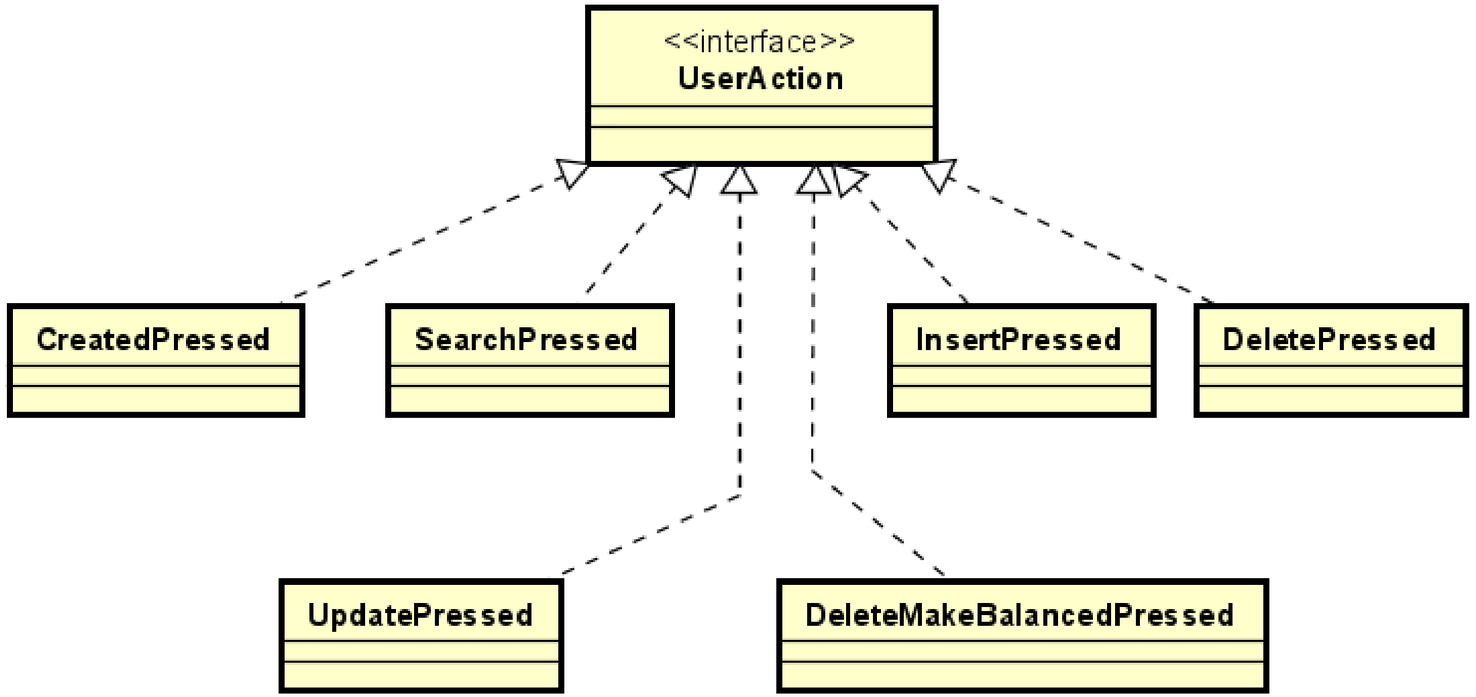
Encapsulation

- We've used the Encapsulation technique in almost all classes to hide the internal state and implementation details from external entities.
- For example, in class Node, we set 'private' for its attributes such as NodeId, depth, …
- If other classes want to access the attributes, they can use getter and setter methods.

4. Explaination of OOP techniques

Polymorphism

Firstly, we have used the Polymorphism technique when implementing an “undo” operation.

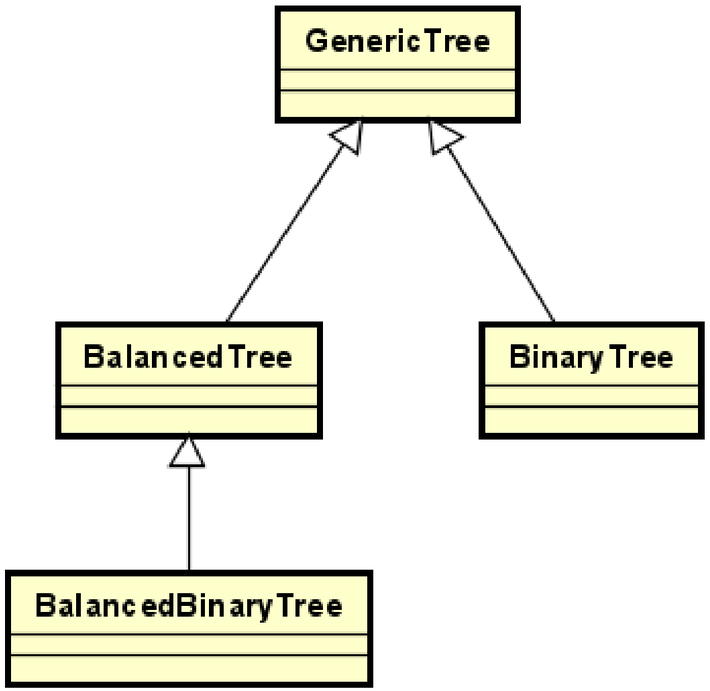


package operation

4. Explaination of OOP techniques

Polymorphism

Secondly, we have used the technique when implementing each operation.



package treedatastructure

4. Explaination of OOP techniques

Association & Aggregation

- GenericTreeController controls and manipulates the GenericTree.
- The operation classes in package operation change the GenericTree and manipulate the tree-display Pane of GenericTreeController.

- The Node has its parent which is also a Node and when the Node is deleted, its parent still exists.
- In addition, this relationship is presented where the GenericTree has a root Node.

4. Explaination of OOP techniques

Composition

- The Node contains three objects Circle, TextField, Line which are used to display the node on screen.
- The Node has many children (which are stored in an attribute arraylist listOfChildren).
- GenericTreeController has an array named history which contains many UserAction-implemented objects.

4. Explaination of OOP techniques

Dependency

- The controller classes depend on the tree data structure classes.
- The GenericTreeController class depends on the operation classes in package operation to perform the corresponding tree operations.
- The main application class depends on the ‘controller’ package.
- In the operations of four types of tree, we also use the Exceptions, so the Trees are dependent on the Exceptions.

References

[1] Stack Overflow: Link: https://stackoverflow.com/

[2] Visualgo - BST Visualization: Link: https://visualgo.net/en/bst

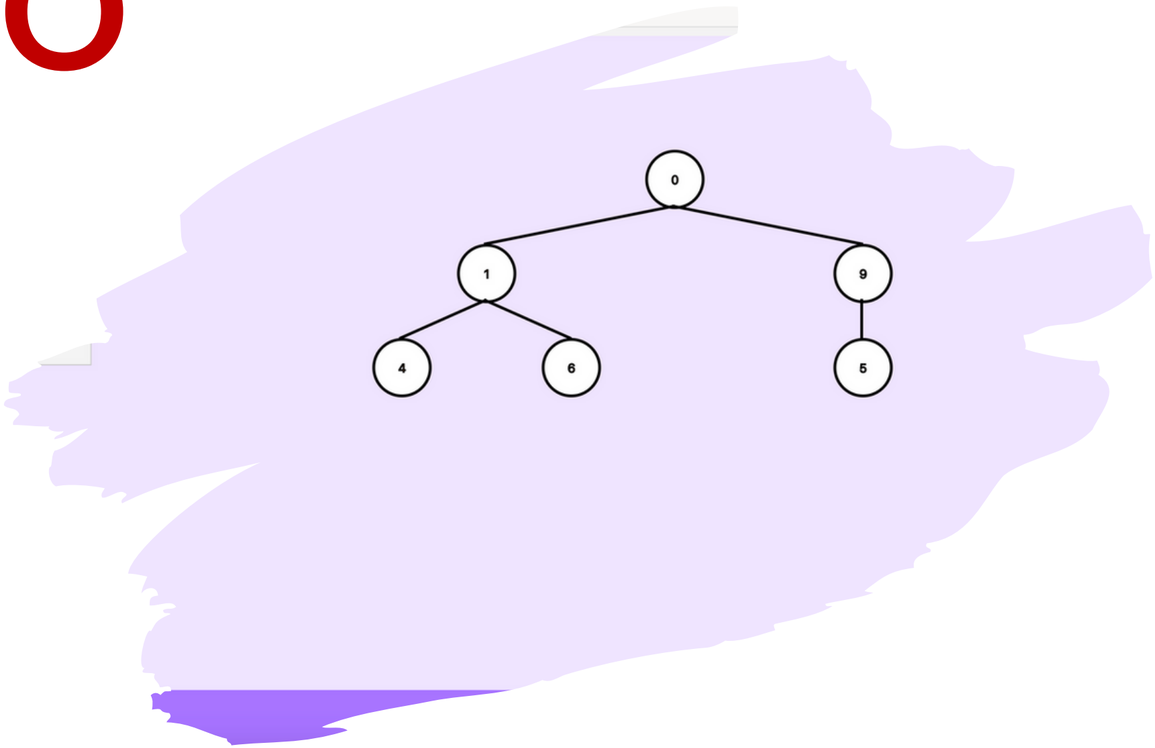
[3] We have consulted about the design of inheritance relationship among tree structures and node layout with the authors, who are our seniors having excellent project implementation.
Link: https://github.com/tamht194450/OOP.DSAI.20202.Team13/tree/main

[4] Insert text to shape
Link: How do I insert text into a shape in Javafx? - Stack Overflow

HUST

hust.edu.vn fb.com/dhbkhn

DEMO



HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !