**Group 13:**

***Project Topic: Visualization of operations on tree data structures***

1. **Basis knowledge:**
   * Generic tree
   * Binary tree
   * Balanced tree
   * Balanced binary tree
   * References:
     + <https://web.stanford.edu/class/archive/cs/cs166/cs166.1146/lectures/02/Small02.pdf>
     + <https://www.programiz.com/dsa/balanced-binary-tree>
     + <https://www.geeksforgeeks.org/generic-treesn-array-trees/>
     + <https://www.pepcoding.com/resources/online-java-foundation/generic-tree>
     + <https://www.geeksforgeeks.org/binary-tree-set-1-introduction/>
     + <https://course.ccs.neu.edu/cs5010f18/lecture9.html>
     + <https://www.programmersought.com/article/59481764947/>
2. **Operations:**
   * Create
   * Insert
   * Delete
   * Update
   * Traverse: preorder, postorder, inorder
   * Search
3. **Use case Diagram**:
   * Actor:
     + Users
   * Use cases:
     + Choose type of tree data structures
     + View introduction and orientation
     + Create a new tree
     + Insert node
     + Delete node
     + Update node
     + Traverse tree
     + Search node
     + Exit
4. General class diagram: (Deadlines)
   * Package: oop.ict.project.tree
   * Classes:
     + Node: 1 - Hằng
     + Generic Tree: 1 - Tối thứ 6
     + Binary Tree: 2 - Vũ - Chủ nhật
     + Balanced Tree: 3 - Hiếu - Chủ nhật
     + Balanced Binary Tree (extends) Balanced Tree: 1
     + Nhớ: tìm hiểu luyện thêm tí javaFx cho quen trong khoảng tg trống nữa nhé.
   * Package: oop.ict.project.gui
     + Screen Menu: display 4 types of tree data structures to choose
     + Screen Operations: display 6 operations to choose
     + Display Screen: visualize each operation as well as the tree data structure.
     + Help Menu
5. **GUI**: JavaFx
   * Some references:
     + <https://github.com/EricCanull/fxbinarytree>
6. **Design**:
   * We only consider undirected-weight trees,with integer node values and no duplicated node values allowed.
   * For the balanced tree and balanced binary tree, the maximum difference in distance from the root of the leaf nodes must be chosen by the user.
   * On the main menu:title of the application,navigation bar for users to choose between the four types of tree,help menu and quit.
     + Users must select a type of data structure before getting into the visualization.
     + The help menu shows basic usage and aim of the project.
     + Quit button exits the application. Remember To ask for confirmation.
   * In the visualization
     + Users can choose to visualize one of six operations, by selecting an option on the operations menu, and then provide the necessary parameter.
     + When an operation starts to execute, on the code panel,the pseudo code (or actual code) should be displayed, and the currently executing line is highlighted to help the user keep track of the process. On the bottom bar, the user can see the progressbar of the executing operation and choose to pause,continue, or go backward or forward a step in the execution.
     + The user can also undo or redo operations from the bottom bar.
     + Always have a Back button for user to return to main menu at any time
7. **Naming convention and Notes:**
   * Package: oop.ict.<project\_name>.<package\_name>
   * Class and Interface: Follow the naming convention as in OOP Lab (For example: CartScreen) - The first character of each word has to be in uppercase.
   * Methods, Constructors and Variables: have to be related to the problems. The beginning character must be in lowercase. For example: void printInorder(), void createBinaryTree(), etc.
   * Constants: in uppercase, each word separated by \_ For example: MAX\_LENGTH
   * Always format code: IntelliJ: Ctrl + Alt + L - Eclipse: Ctrl + Shift + F
   * Java version: 8, 11 or 15? Java 11: stable and most recent version featuring long term support; Java 15: offer some new features, or Java 8. => Java 15
   * Comments:
     + Each class must have comments about it.
     + Complicated methods have to have comments about their function.
     + Brief comment for each variable.
   * Some other notes:
     + Short code => One code - one line.
     + Code too long => Remember this white **straight** line. Don’t let the code exceed it.
   * Git: Đây là quy trình dùng Git lên khi làm việc nhóm mà t biết được chút từ các anh:
     + Clone Project from git to your local repo
     + For each feature, you can checkout to a new branch. When finish, push to this newly created branch and merge to master(create merge request) - Đoạn này t chưa merge lần nào nên cũng chưa nắm thực sự chắc chắn “https://freetuts.net/git-lenh-merge-branch-xu-ly-conflict-1076.html#:~:text=Merge%20branch%20t%E1%BB%A9c%20l%C3%A0%20b%E1%BA%A1n,gi%E1%BA%A3i%20quy%E1%BA%BFt%20chung%20m%E1%BB%99t%20task.”
     + Giả sử t đang code feature binary tree chẳng hạn mà Hằng hoặc Hiếu code xong phần Generic tree, push lên và merge rồi. Thì trước khi t push lên branch mới của t, t sẽ fai git pull master về để xem có conflict code gì không.
     + Sau đó mình mới push lên branch features của mình.
     + git fetch: để lấy các branch mới (nếu có)
     + git checkout master & git pull origin master: để lấy code mới nhất từ master
     + git checkout -b new\_branch: tạo branch mới để làm việc
     + commit xong thì tạo pull request để merge code vào master.

