Lab 2

In this lab, you will extend a codebase to incorporate a Binary Search Tree (BST). Your BST must implement all methods so that both the main program **and** the BSTTester work.

Setup

1. Download the base project and open it in IntelliJ.
2. Change the package to incorporate your name (edu.ncssm. `username`)
3. Read through the code and investigate the BSTree class. You will need to implement all the public methods.
4. Add appropriate JavaDoc and comments throughout the BSTree class

Implementation Details

- I have declared all the public methods and inner classes needed for this lab. You are not permitted to alter any public signatures without permission. You may add any private elements you need.
- **NEVER** return a `BSTNode` or have it as a parameter type of public method.
- BSTs are pretty much optimized, so you should have to reduce duplicate code.
- All non-public methods must have proper JavaDoc comments, and you must include comments with code that is not self-documenting. I have purposefully not commented the BSTree file.

Rubric (Base Tier)

- Style/Documentation
  - Variable, Method, and Class names
  - Commenting and JavaDoc
- Method Implementation
  - Constructors - Use Natural Ordering unless a Comparator is provided.
  - `add` - Correctly insert data into the BST

- `remove` - Remove the **first** instance of data from the BST (handles zero, one, and two children)
- `removeAll` - Remove **all** instances of data from the BST
- Traversals - Pre-, Post-, and In-Order traversals must work

Additional Tiers (Complete 2 of 3 for a Late Day)

1. Add the ability to handle global key events:
    i. ESC - deselect all objects
    ii. Delete - delete the selected objects
2. Add the ability to move the selected shape
    i. Arrow keys
    ii. Click a drag with the mouse
3. Add a custom tool (rectangle/circle) and provide a way to switch between tools