Binary Search Tree

Problem Statement

You are required to implement a binary search tree (BST) in Java. The provided skeleton code outlines the basic structure for the BST, including the `Node` class that represents each node in the tree, and various methods to manipulate and traverse the tree. Your task is to complete the implementation and enhance it with additional features.

Tasks:

1. Implement the Constructor: Complete the constructor `BinarySearchTree()` to initialize an empty binary search tree.
2. Implement `insert(int data)` Method: Write the `insert(int data)` method to add a new node with the specified data into the binary search tree. Ensure the tree maintains its binary search property.
3. Implement `inorder()` Method: Write the `inorder()` method to perform an in-order traversal of the binary search tree and print the elements in sorted order.
4. Implement `preorder()` Method: Write the `preorder()` method to perform a pre-order traversal of the binary search tree.
5. Implement `postorder()` Method: Write the `postorder()` method to perform a post-order traversal of the binary search tree.

Additional Tasks:

1. Use Generics: Modify the implementation to use generics, allowing the binary search tree to store elements of any comparable type.
2. Implement `find(int data)` Method: Add a method `find(int data)` to search for a specific value in the binary search tree and return the node containing that value.
3. Implement `delete(int data)` Method: Write a method `delete(int data)` to remove a node with the specified value from the binary search tree. Handle all cases, including nodes with two children.
4. Implement `findMin()` and `findMax()` Methods: Write methods to find and return the minimum and maximum values in the binary search tree.
5. Implement `height()` Method: Add a method to calculate and return the height of the binary search tree.
6. Check for Balanced Tree: Implement a method to check whether the binary search tree is balanced (i.e., the difference in height between the left and right subtrees of any node is no more than one).
7. Implement `isBST()` Method: Write a method to check if a given binary tree is a valid binary search tree.