Test cases design

Stages

Name	Class	Stage
setupStage1	MasterClass	readFiles() reads the file at test/data/testData.csv
setup1	AVL	Initialize AVL<Integer, Integer>.
setup2	AVL	10 values are added to the tree such that there is an element 7 with key 15 and there is not an element with key 32.
setup1	BST	Initialize BST<Character, Integer>
setup2	BST	Add t, 305 r, 35 h, 23
setup1	Node	Initialize Node<Character, Integer>
setup1	AVLNode	Initialize AVLNode<Integer, Integer>(10, 1, null)
setup1	RedBlackTree	Initialize RedBlackTree

Test cases

Class	Method	Stage	Input values	Output or result
MasterClass	readFile	setupStage1	testData.csv	Every tree in MasterClass has left and right children sprouting from the root.
MasterClass	search	setupStage1	TRB, 7	The result of search() is saved in an ArrayList whose size is greater than zero.
AVL	add	setup1	10, 20 11, 20 12, 20 13, 20	The size of the tree is 1 after the first addition. The root is balanced, the weight of the root is 3 and the key of the left child of the root is 10.
AVL	remove	setup1 setup2	15	The size of the tree is 10 and the value 7 is found when searching using the key 15. Then, it

				cannot be found after being removed.
AVL	search	setup1 setup2	12 32	The value 7 is found when searching using the key 15. No value is found using the key 32.
AVL	isEmpty	setup1	-	The tree is empty before adding anything, and it stops being so after that.
AVL	keyExists	setup1	6 2	Added keys exist and the opposite.
BST	add	setup1	a, 5 A, 4 z, 8	The size is 0 in the beginning. Then, the size is 3 and searching A returns 4.
BST	search	setup1 setup2	t, 305 r, 35 h, 23	Existing values can be found using their keys, while nonexistent keys return null.
BST	remove	setup1 setup2	H t h	Can't remove H. Size is 2 after removing t and search doesn't find it. The key to the root is r. After adding t, 305 and removing t, the size is 2 and the root has no left child.
BST	isEmpty	setup1	-	The tree is empty before adding anything, and it stops being so after that.
BST	keyExists	setup1	b f	Added keys exist and the opposite.
Node	add	setup1 setup2	-	Right value is 5 and left is 30. Original node is parent of right.
Node	search	setup1 setup2	c	Key c returns 12.
Node	remove	setup1 setup2	b	Value of right is 12

AVLNode	add	setup1	7, 2 12, 3	Nodes are added correctly and weight changes accordingly.
AVLNode	remove	setup1	12 7	Nodes are removed correctly and weight changes accordingly.
AVLNode	search	setup1	24	Added values are subsequently found.
AVLNode	leftRotate	setup1	-	Nodes are repositioned to the left and weight is adjusted accordingly.
AVLNode	rightRotate	setup1	-	Nodes are repositioned to the right and weight is adjusted accordingly.
AVLNode	balance	setup1	-	isBalance becomes true after balancing.
RedBlackTree	insert	setup1	various	Added nodes are found and the opposite.
RedBlackTree	searchValue	setup1	12 6 21	Same as insert test.