GTU Department of Computer Engineering CSE 222/505 - Spring 2021

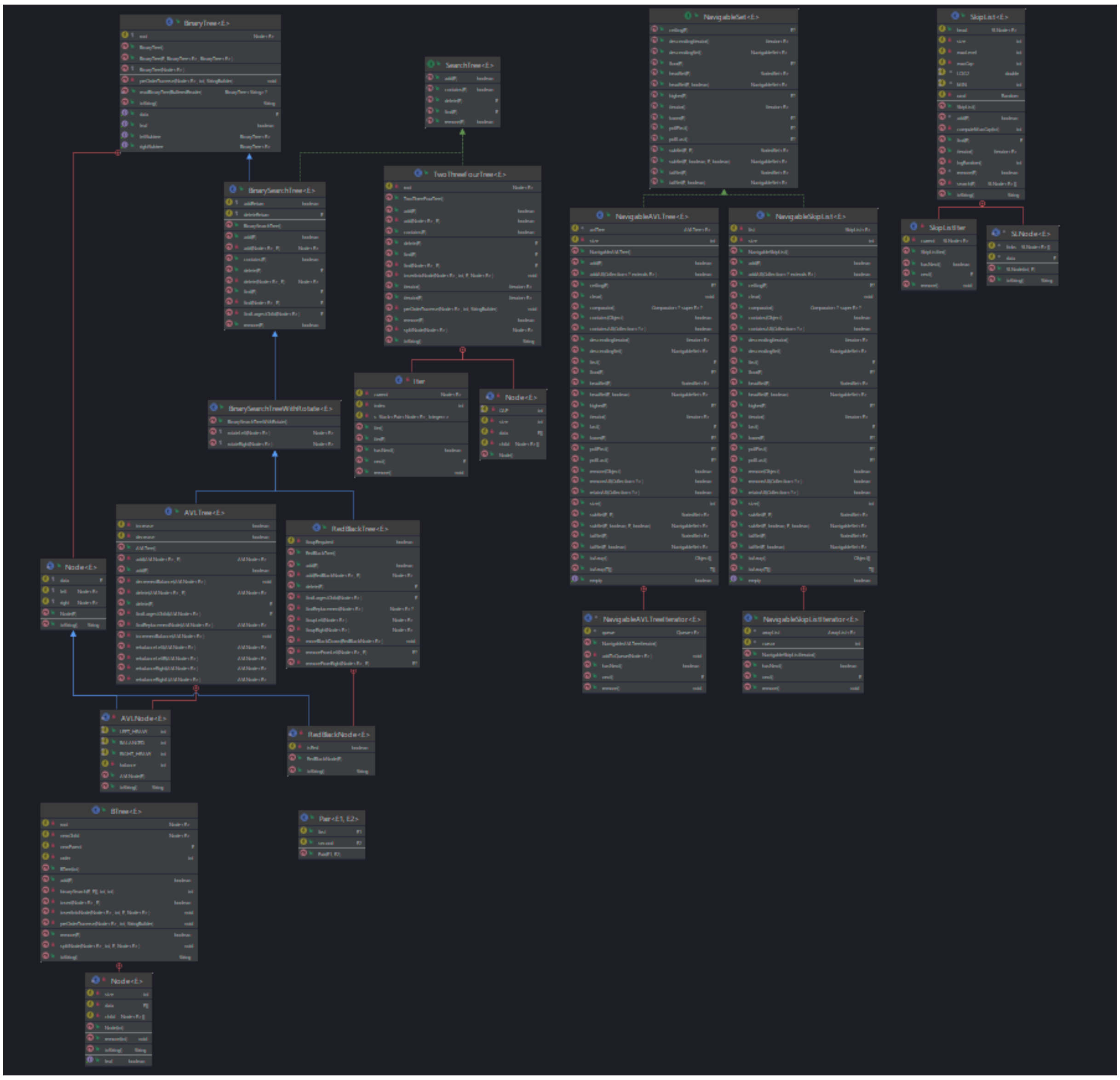
Homework 7 Report

BAHADIR
ETKA KILINÇ
1901042701

Problem Solutions Approach

In this project, I fulfilled the requested tasks under three main headings. First of all, I wrote the avltree and skiplist classes that implement the desired navigableset in part1. I used has-A relation while writing these. In part2, I wrote a method that finds the type of tree sent. In part 3, I completed the project by inserting the desired number of items into the desired data structures and measuring their duration.

Class Diagram

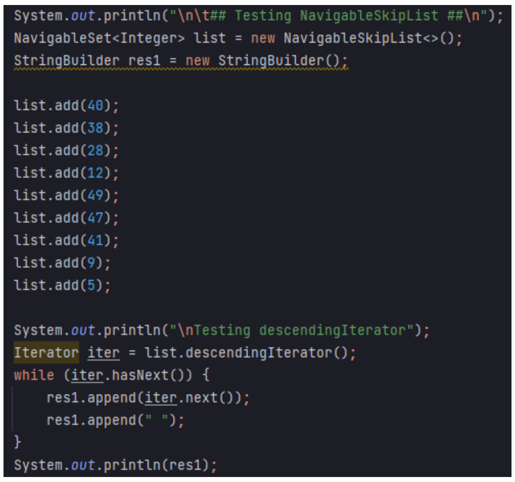


Running and Results

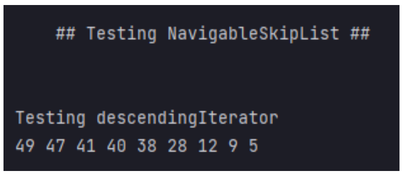
Part 1

Creating an NavigableSkipList with using NavigableSet Interface.

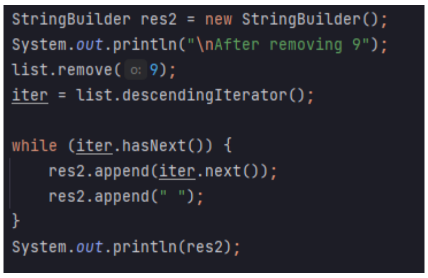
Testing add,remove and NavigableSkipList's descendingIterator.



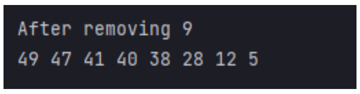
```
System.out.println("\n\t## Testing NavigableSkipList ##\n");
NavigableSet<Integer> list = new NavigableSkipList<>();
StringBuilder res1 = new StringBuilder();

list.add(40);
list.add(38);
list.add(28);
list.add(12);
list.add(49);
list.add(47);
list.add(41);
list.add(9);
list.add(5);

System.out.println("\nTesting descendingIterator");
Iterator iter = list.descendingIterator();
while (iter.hasNext()) {
    res1.append(iter.next());
    res1.append(" ");
}
System.out.println(res1);
```



```
    ## Testing NavigableSkipList ##


Testing descendingIterator
49 47 41 40 38 28 12 9 5
```



```
StringBuilder res2 = new StringBuilder();
System.out.println("\nAfter removing 9");
list.remove( o: 9);
iter = list.descendingIterator();

while (iter.hasNext()) {
    res2.append(iter.next());
    res2.append(" ");
}
System.out.println(res2);
```

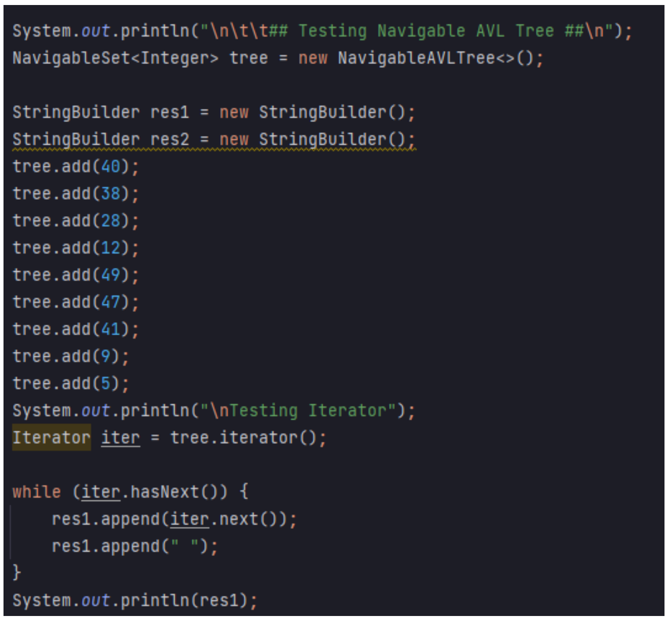


```
After removing 9
49 47 41 40 38 28 12 5
```

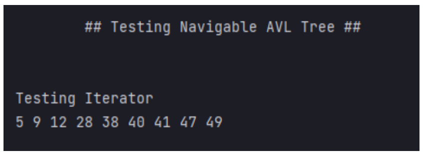
Part 2

Creating NavigableAVLTree with using NavigableSet interface.

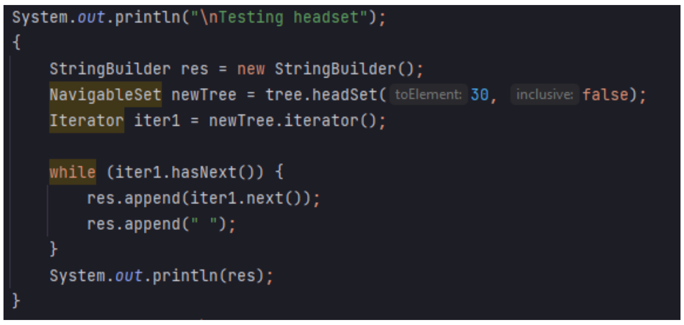
Testing add, remove, iterator, headSet, tailSet.



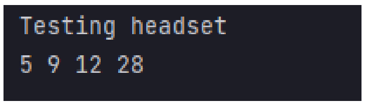
```java
System.out.println("\n\t\t## Testing Navigable AVL Tree ##\n");
NavigableSet<Integer> tree = new NavigableAVLTree<>();

StringBuilder res1 = new StringBuilder();
StringBuilder res2 = new StringBuilder();
tree.add(40);
tree.add(38);
tree.add(28);
tree.add(12);
tree.add(49);
tree.add(47);
tree.add(41);
tree.add(9);
tree.add(5);
System.out.println("\nTesting Iterator");
Iterator iter = tree.iterator();

while (iter.hasNext()) {
    res1.append(iter.next());
    res1.append(" ");
}
System.out.println(res1);
```



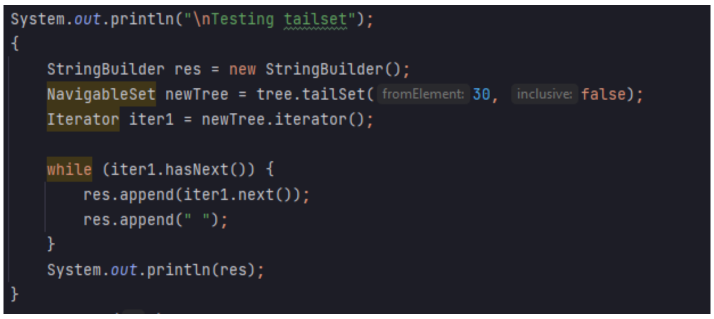
```
        ## Testing Navigable AVL Tree ##


Testing Iterator
5 9 12 28 38 40 41 47 49
```



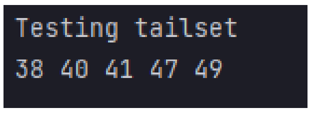
```java
System.out.println("\nTesting headset");
{
    StringBuilder res = new StringBuilder();
    NavigableSet newTree = tree.headSet( toElement: 30,  inclusive: false);
    Iterator iter1 = newTree.iterator();

    while (iter1.hasNext()) {
        res.append(iter1.next());
        res.append(" ");
    }
    System.out.println(res);
}
```

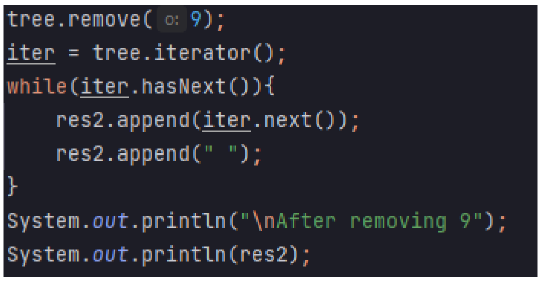


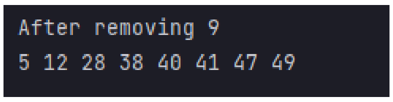
```
Testing headset
5 9 12 28
```



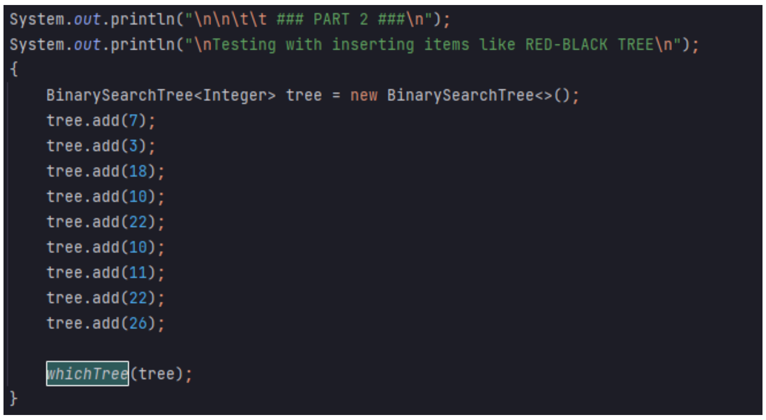
```java
System.out.println("\nTesting tailset");
{
    StringBuilder res = new StringBuilder();
    NavigableSet newTree = tree.tailSet( fromElement: 30,  inclusive: false);
    Iterator iter1 = newTree.iterator();

    while (iter1.hasNext()) {
        res.append(iter1.next());
        res.append(" ");
    }
    System.out.println(res);
}
```

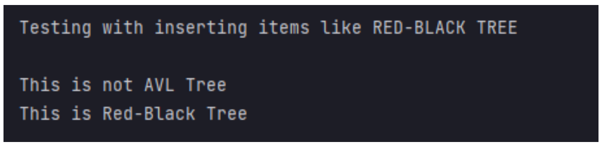

```
Testing tailset
38 40 41 47 49
```

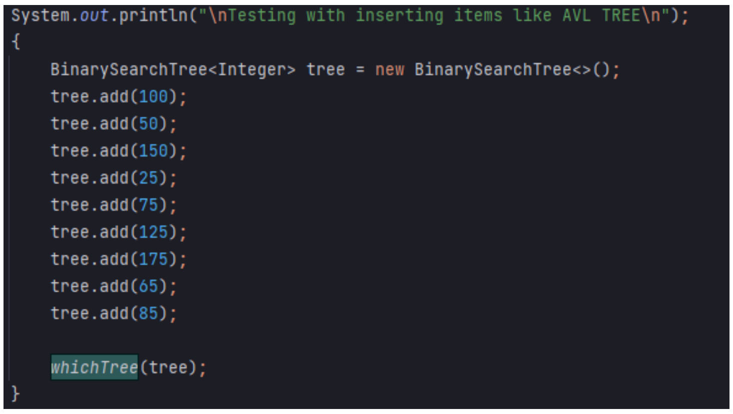

```
tree.remove( o: 9);
iter = tree.iterator();
while(iter.hasNext()){
    res2.append(iter.next());
    res2.append(" ");
}
System.out.println("\nAfter removing 9");
System.out.println(res2);
```

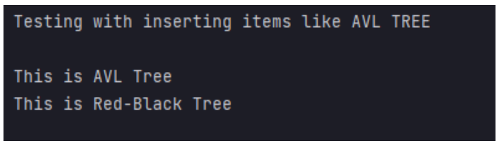

```
After removing 9
5 12 28 38 40 41 47 49
```

Creating a binary search tree and testing which tree is(red-black or avl)


```
System.out.println("\n\n\t\t ### PART 2 ###\n");
System.out.println("\nTesting with inserting items like RED-BLACK TREE\n");
{
    BinarySearchTree<Integer> tree = new BinarySearchTree<>();
    tree.add(7);
    tree.add(3);
    tree.add(18);
    tree.add(10);
    tree.add(22);
    tree.add(10);
    tree.add(11);
    tree.add(22);
    tree.add(26);

    whichTree(tree);
}
```

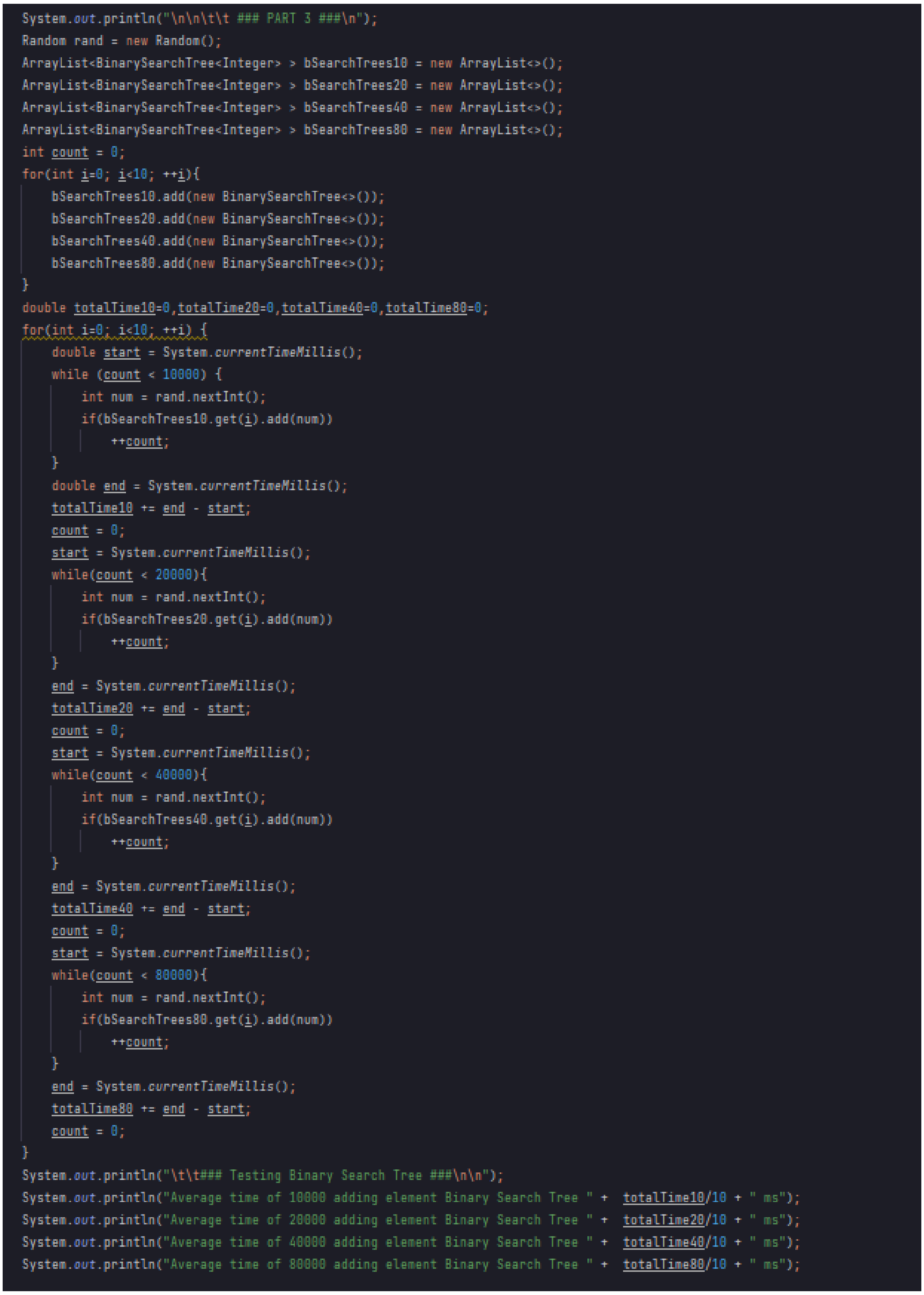

```
Testing with inserting items like RED-BLACK TREE

This is not AVL Tree
This is Red-Black Tree
```


```
System.out.println("\nTesting with inserting items like AVL TREE\n");
{
    BinarySearchTree<Integer> tree = new BinarySearchTree<>();
    tree.add(100);
    tree.add(50);
    tree.add(150);
    tree.add(25);
    tree.add(75);
    tree.add(125);
    tree.add(175);
    tree.add(65);
    tree.add(85);

    whichTree(tree);
}
```

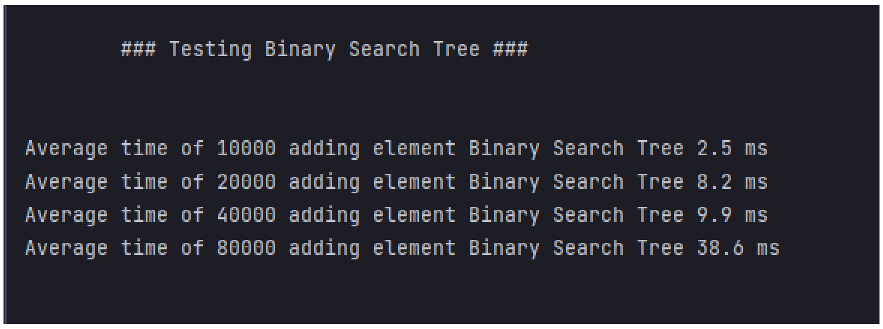

```
Testing with inserting items like AVL TREE

This is AVL Tree
This is Red-Black Tree
```

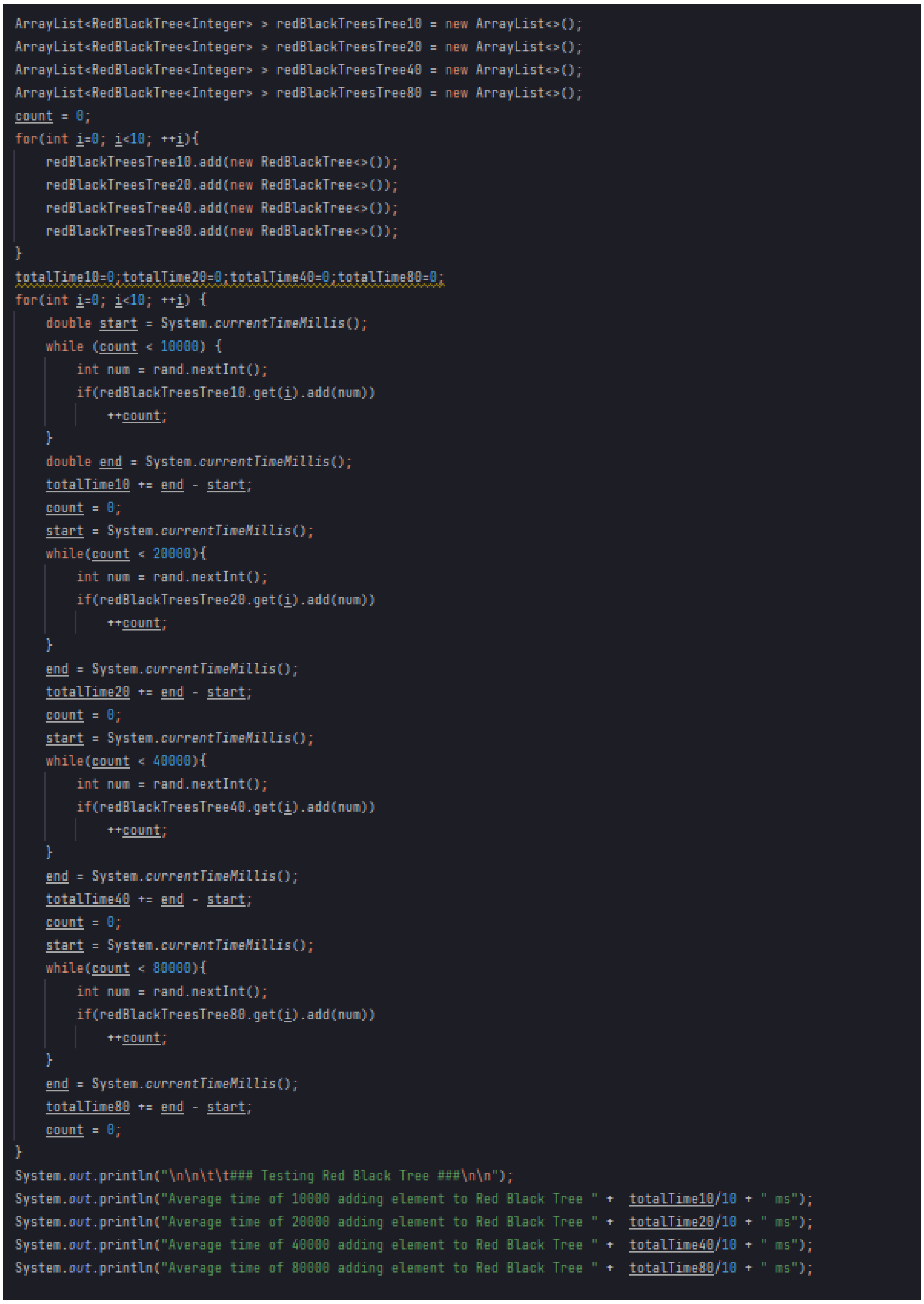
Part 3

Testing required data structures with a random number and measuring running time.

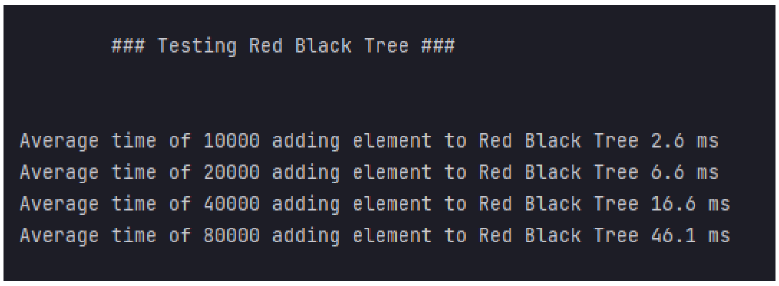


```java
System.out.println("\n\n\t\t ### PART 3 ###\n");
Random rand = new Random();
ArrayList<BinarySearchTree<Integer> > bSearchTrees10 = new ArrayList<>();
ArrayList<BinarySearchTree<Integer> > bSearchTrees20 = new ArrayList<>();
ArrayList<BinarySearchTree<Integer> > bSearchTrees40 = new ArrayList<>();
ArrayList<BinarySearchTree<Integer> > bSearchTrees80 = new ArrayList<>();
int count = 0;
for(int i=0; i<10; ++i){
    bSearchTrees10.add(new BinarySearchTree<>());
    bSearchTrees20.add(new BinarySearchTree<>());
    bSearchTrees40.add(new BinarySearchTree<>());
    bSearchTrees80.add(new BinarySearchTree<>());
}
double totalTime10=0,totalTime20=0,totalTime40=0,totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.currentTimeMillis();
    while (count < 10000) {
        int num = rand.nextInt();
        if(bSearchTrees10.get(i).add(num))
            ++count;
    }
    double end = System.currentTimeMillis();
    totalTime10 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 20000){
        int num = rand.nextInt();
        if(bSearchTrees20.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime20 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 40000){
        int num = rand.nextInt();
        if(bSearchTrees40.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime40 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 80000){
        int num = rand.nextInt();
        if(bSearchTrees80.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime80 += end - start;
    count = 0;
}
System.out.println("\t\t### Testing Binary Search Tree ###\n\n");
System.out.println("Average time of 10000 adding element Binary Search Tree " +  totalTime10/10 + " ms");
System.out.println("Average time of 20000 adding element Binary Search Tree " +  totalTime20/10 + " ms");
System.out.println("Average time of 40000 adding element Binary Search Tree " +  totalTime40/10 + " ms");
System.out.println("Average time of 80000 adding element Binary Search Tree " +  totalTime80/10 + " ms");
```

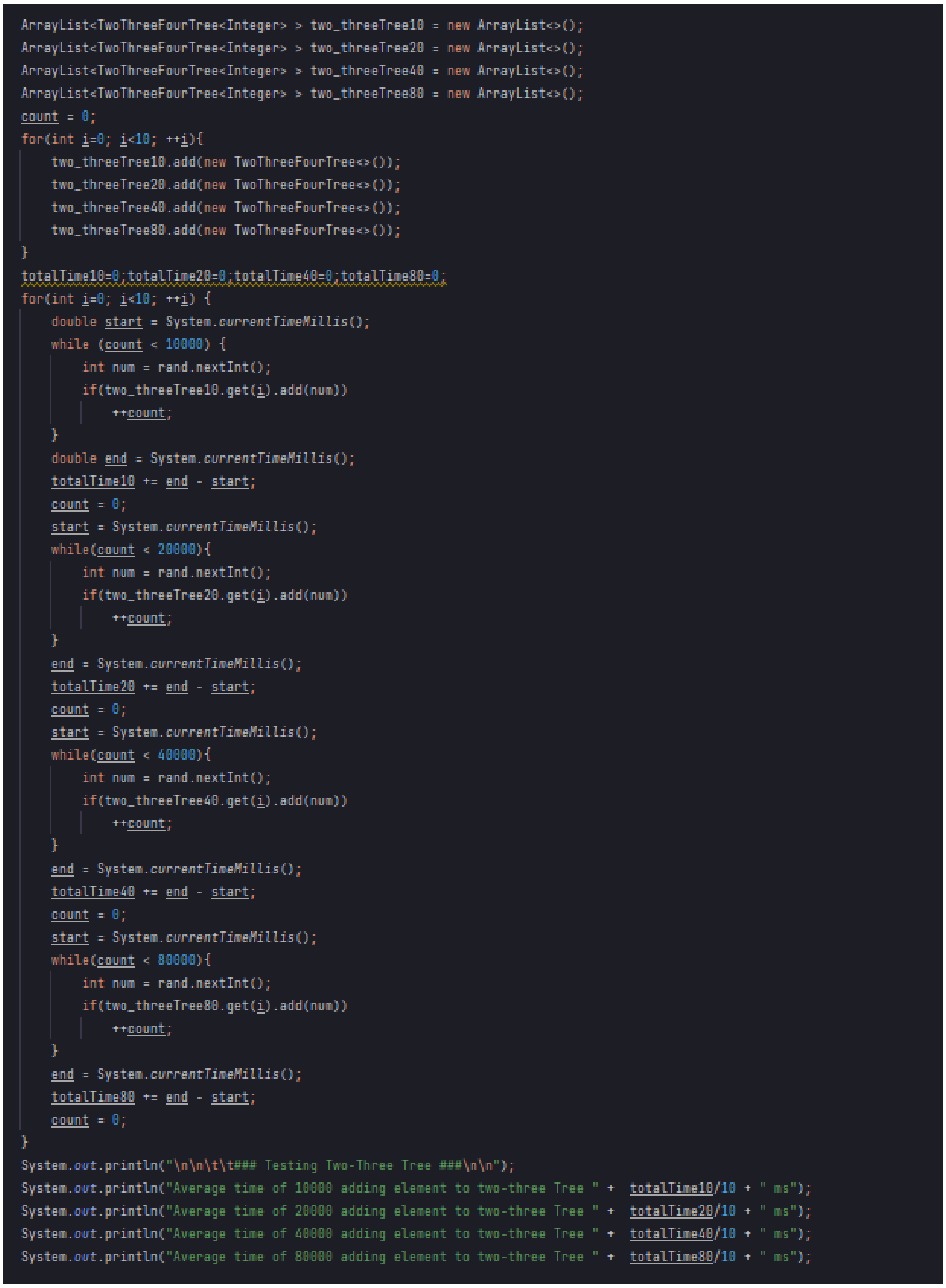

```
        ### Testing Binary Search Tree ###


Average time of 10000 adding element Binary Search Tree 2.5 ms
Average time of 20000 adding element Binary Search Tree 8.2 ms
Average time of 40000 adding element Binary Search Tree 9.9 ms
Average time of 80000 adding element Binary Search Tree 38.6 ms
```



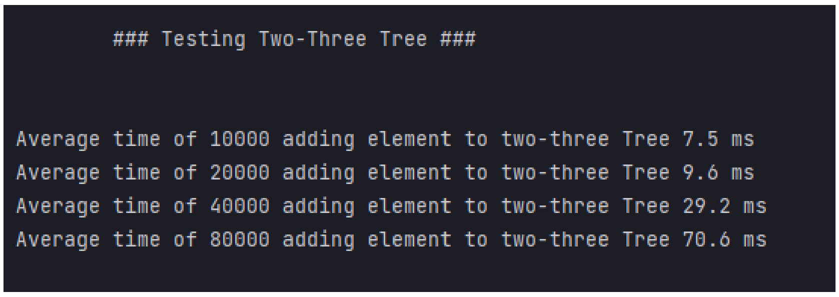
```java
ArrayList<RedBlackTree<Integer> > redBlackTreesTree10 = new ArrayList<>();
ArrayList<RedBlackTree<Integer> > redBlackTreesTree20 = new ArrayList<>();
ArrayList<RedBlackTree<Integer> > redBlackTreesTree40 = new ArrayList<>();
ArrayList<RedBlackTree<Integer> > redBlackTreesTree80 = new ArrayList<>();
count = 0;
for(int i=0; i<10; ++i){
    redBlackTreesTree10.add(new RedBlackTree<>());
    redBlackTreesTree20.add(new RedBlackTree<>());
    redBlackTreesTree40.add(new RedBlackTree<>());
    redBlackTreesTree80.add(new RedBlackTree<>());
}
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.currentTimeMillis();
    while (count < 10000) {
        int num = rand.nextInt();
        if(redBlackTreesTree10.get(i).add(num))
            ++count;
    }
    double end = System.currentTimeMillis();
    totalTime10 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 20000){
        int num = rand.nextInt();
        if(redBlackTreesTree20.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime20 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 40000){
        int num = rand.nextInt();
        if(redBlackTreesTree40.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime40 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 80000){
        int num = rand.nextInt();
        if(redBlackTreesTree80.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime80 += end - start;
    count = 0;
}
System.out.println("\n\n\t\t### Testing Red Black Tree ###\n\n");
System.out.println("Average time of 10000 adding element to Red Black Tree " +  totalTime10/10 + " ms");
System.out.println("Average time of 20000 adding element to Red Black Tree " +  totalTime20/10 + " ms");
System.out.println("Average time of 40000 adding element to Red Black Tree " +  totalTime40/10 + " ms");
System.out.println("Average time of 80000 adding element to Red Black Tree " +  totalTime80/10 + " ms");
```



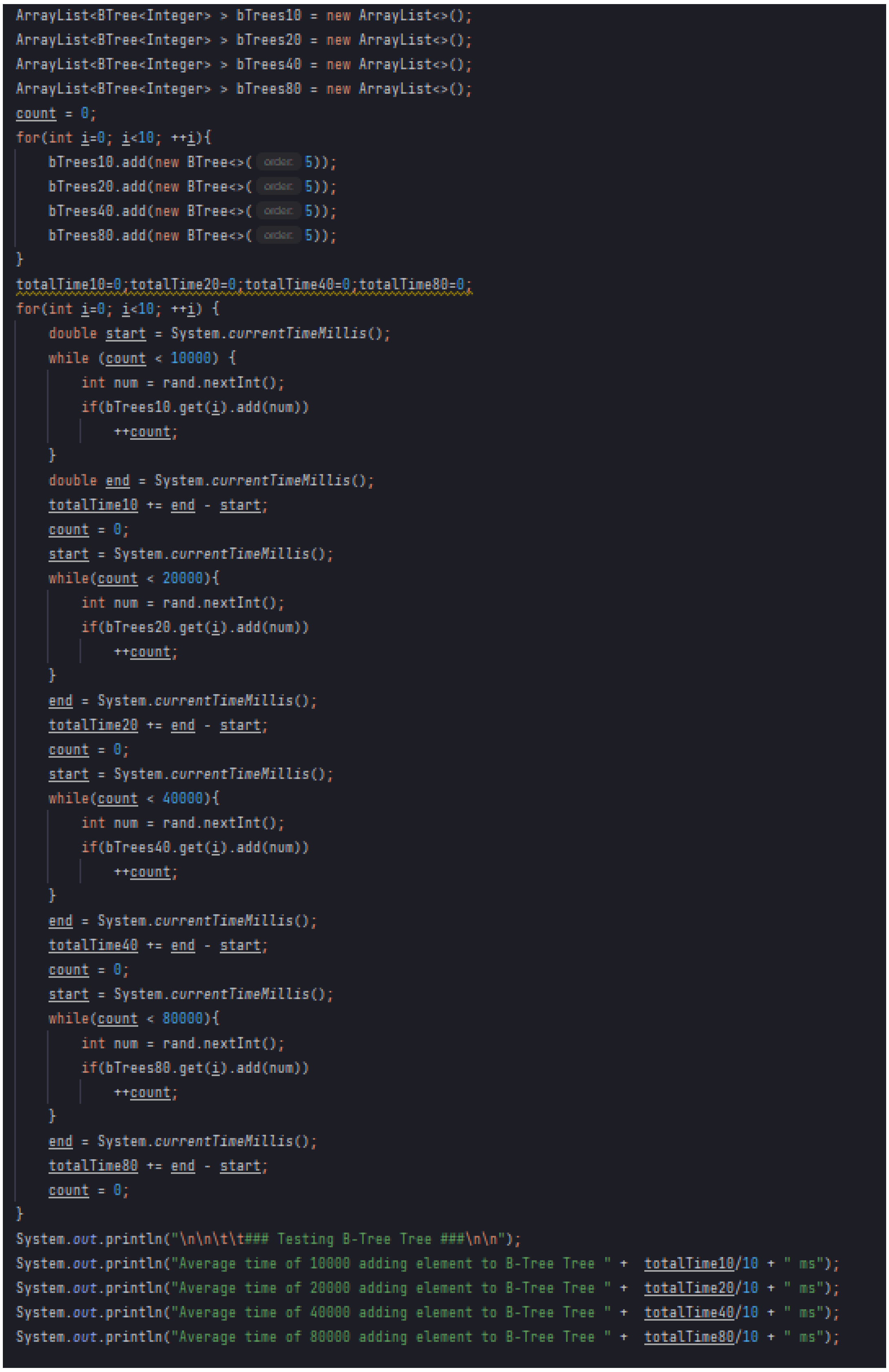
```
        ### Testing Red Black Tree ###


Average time of 10000 adding element to Red Black Tree 2.6 ms
Average time of 20000 adding element to Red Black Tree 6.6 ms
Average time of 40000 adding element to Red Black Tree 16.6 ms
Average time of 80000 adding element to Red Black Tree 46.1 ms
```



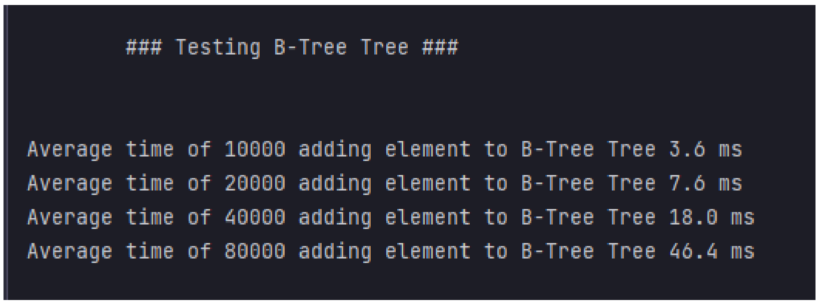
```java
ArrayList<TwoThreeFourTree<Integer> > two_threeTree10 = new ArrayList<>();
ArrayList<TwoThreeFourTree<Integer> > two_threeTree20 = new ArrayList<>();
ArrayList<TwoThreeFourTree<Integer> > two_threeTree40 = new ArrayList<>();
ArrayList<TwoThreeFourTree<Integer> > two_threeTree80 = new ArrayList<>();
count = 0;
for(int i=0; i<10; ++i){
    two_threeTree10.add(new TwoThreeFourTree<>());
    two_threeTree20.add(new TwoThreeFourTree<>());
    two_threeTree40.add(new TwoThreeFourTree<>());
    two_threeTree80.add(new TwoThreeFourTree<>());
}
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.currentTimeMillis();
    while (count < 10000) {
        int num = rand.nextInt();
        if(two_threeTree10.get(i).add(num))
            ++count;
    }
    double end = System.currentTimeMillis();
    totalTime10 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 20000){
        int num = rand.nextInt();
        if(two_threeTree20.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime20 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 40000){
        int num = rand.nextInt();
        if(two_threeTree40.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime40 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 80000){
        int num = rand.nextInt();
        if(two_threeTree80.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime80 += end - start;
    count = 0;
}
System.out.println("\n\n\t\t### Testing Two-Three Tree ###\n\n");
System.out.println("Average time of 10000 adding element to two-three Tree " +  totalTime10/10 + " ms");
System.out.println("Average time of 20000 adding element to two-three Tree " +  totalTime20/10 + " ms");
System.out.println("Average time of 40000 adding element to two-three Tree " +  totalTime40/10 + " ms");
System.out.println("Average time of 80000 adding element to two-three Tree " +  totalTime80/10 + " ms");
```



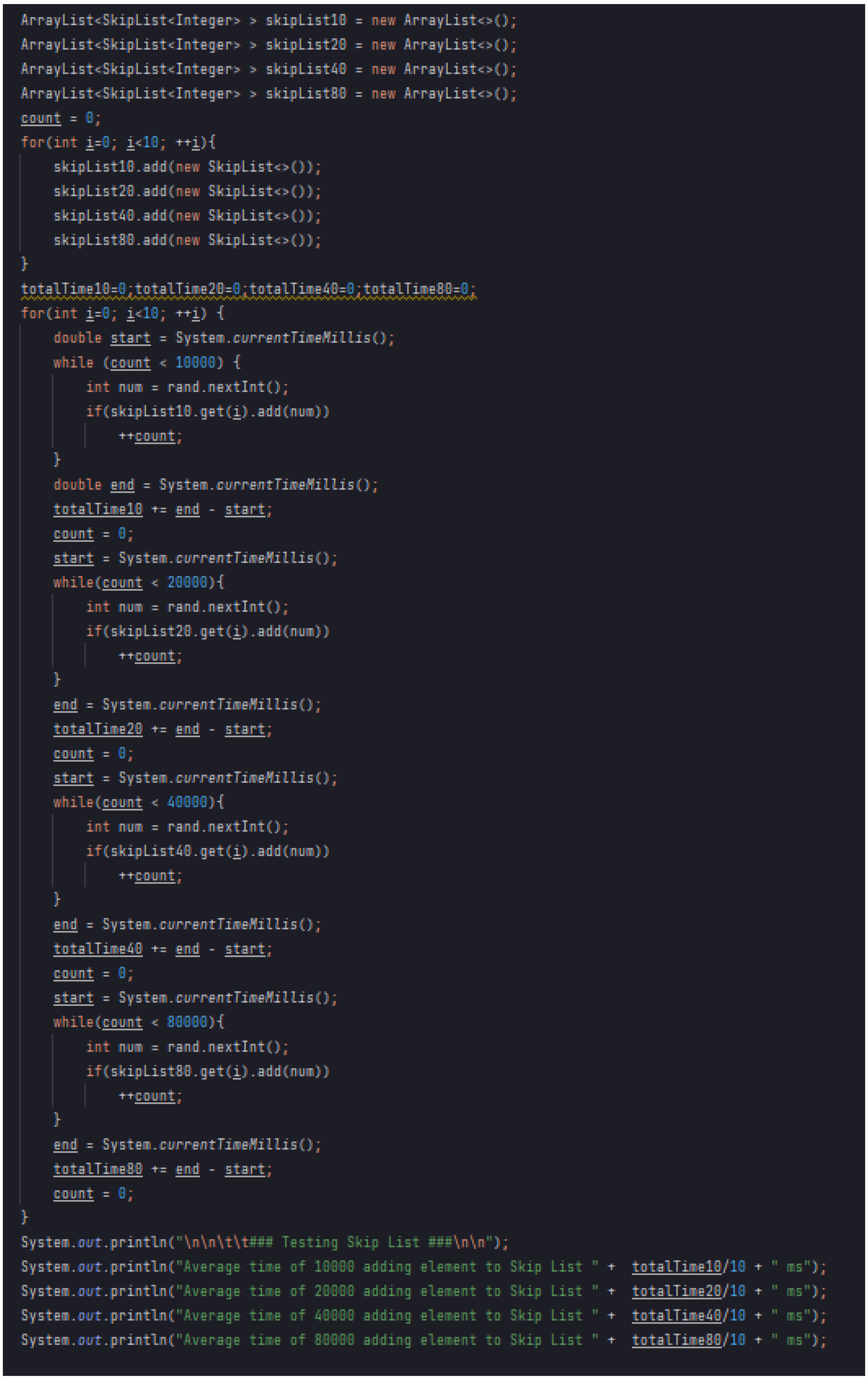
```
        ### Testing Two-Three Tree ###


Average time of 10000 adding element to two-three Tree 7.5 ms
Average time of 20000 adding element to two-three Tree 9.6 ms
Average time of 40000 adding element to two-three Tree 29.2 ms
Average time of 80000 adding element to two-three Tree 70.6 ms
```



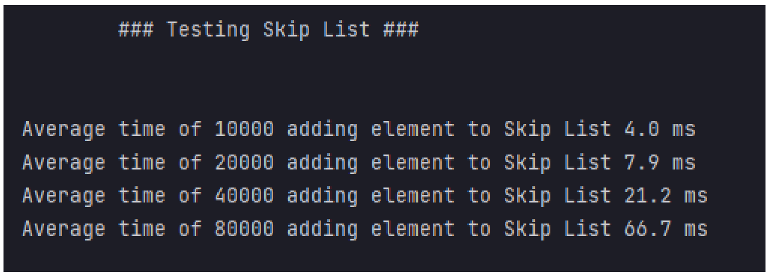
```java
ArrayList<BTree<Integer> > bTrees10 = new ArrayList<>();
ArrayList<BTree<Integer> > bTrees20 = new ArrayList<>();
ArrayList<BTree<Integer> > bTrees40 = new ArrayList<>();
ArrayList<BTree<Integer> > bTrees80 = new ArrayList<>();
count = 0;
for(int i=0; i<10; ++i){
    bTrees10.add(new BTree<>( order: 5));
    bTrees20.add(new BTree<>( order: 5));
    bTrees40.add(new BTree<>( order: 5));
    bTrees80.add(new BTree<>( order: 5));
}
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.currentTimeMillis();
    while (count < 10000) {
        int num = rand.nextInt();
        if(bTrees10.get(i).add(num))
            ++count;
    }
    double end = System.currentTimeMillis();
    totalTime10 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 20000){
        int num = rand.nextInt();
        if(bTrees20.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime20 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 40000){
        int num = rand.nextInt();
        if(bTrees40.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime40 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 80000){
        int num = rand.nextInt();
        if(bTrees80.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime80 += end - start;
    count = 0;
}
System.out.println("\n\n\t\t### Testing B-Tree Tree ###\n\n");
System.out.println("Average time of 10000 adding element to B-Tree Tree " +  totalTime10/10 + " ms");
System.out.println("Average time of 20000 adding element to B-Tree Tree " +  totalTime20/10 + " ms");
System.out.println("Average time of 40000 adding element to B-Tree Tree " +  totalTime40/10 + " ms");
System.out.println("Average time of 80000 adding element to B-Tree Tree " +  totalTime80/10 + " ms");
```



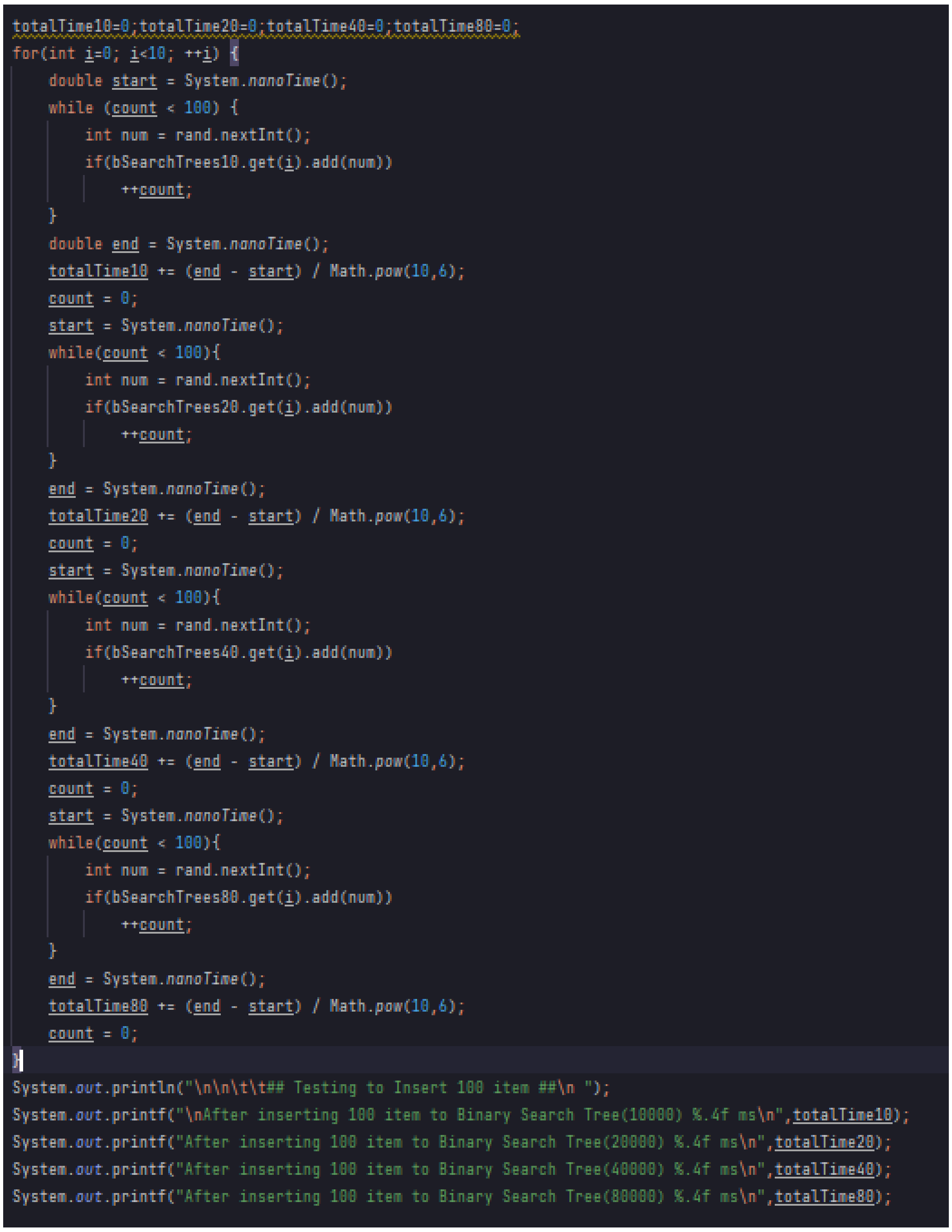
```
        ### Testing B-Tree Tree ###


Average time of 10000 adding element to B-Tree Tree 3.6 ms
Average time of 20000 adding element to B-Tree Tree 7.6 ms
Average time of 40000 adding element to B-Tree Tree 18.0 ms
Average time of 80000 adding element to B-Tree Tree 46.4 ms
```



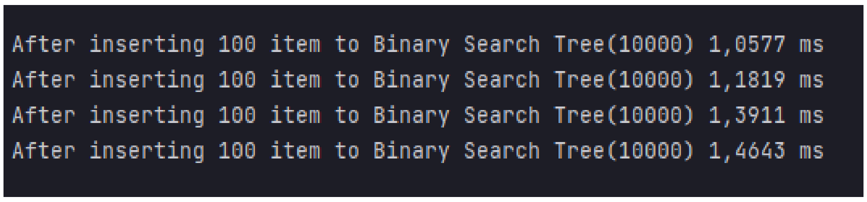
```java
ArrayList<SkipList<Integer> > skipList10 = new ArrayList<>();
ArrayList<SkipList<Integer> > skipList20 = new ArrayList<>();
ArrayList<SkipList<Integer> > skipList40 = new ArrayList<>();
ArrayList<SkipList<Integer> > skipList80 = new ArrayList<>();
count = 0;
for(int i=0; i<10; ++i){
    skipList10.add(new SkipList<>());
    skipList20.add(new SkipList<>());
    skipList40.add(new SkipList<>());
    skipList80.add(new SkipList<>());
}
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.currentTimeMillis();
    while (count < 10000) {
        int num = rand.nextInt();
        if(skipList10.get(i).add(num))
            ++count;
    }
    double end = System.currentTimeMillis();
    totalTime10 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 20000){
        int num = rand.nextInt();
        if(skipList20.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime20 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 40000){
        int num = rand.nextInt();
        if(skipList40.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime40 += end - start;
    count = 0;
    start = System.currentTimeMillis();
    while(count < 80000){
        int num = rand.nextInt();
        if(skipList80.get(i).add(num))
            ++count;
    }
    end = System.currentTimeMillis();
    totalTime80 += end - start;
    count = 0;
}
System.out.println("\n\n\t\t### Testing Skip List ###\n\n");
System.out.println("Average time of 10000 adding element to Skip List " +  totalTime10/10 + " ms");
System.out.println("Average time of 20000 adding element to Skip List " +  totalTime20/10 + " ms");
System.out.println("Average time of 40000 adding element to Skip List " +  totalTime40/10 + " ms");
System.out.println("Average time of 80000 adding element to Skip List " +  totalTime80/10 + " ms");
```



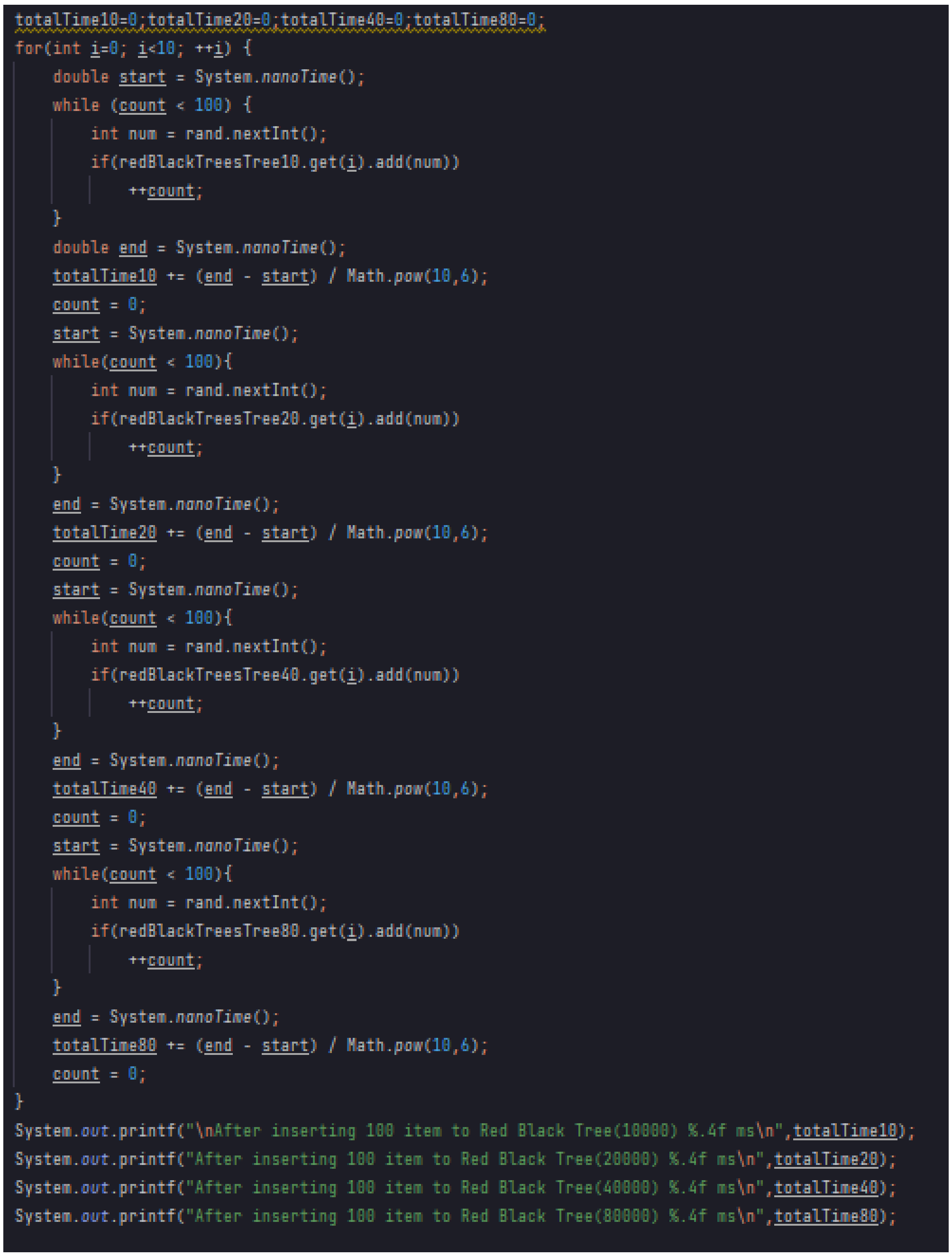
```
         ### Testing Skip List ###


 Average time of 10000 adding element to Skip List 4.0 ms
 Average time of 20000 adding element to Skip List 7.9 ms
 Average time of 40000 adding element to Skip List 21.2 ms
 Average time of 80000 adding element to Skip List 66.7 ms
```



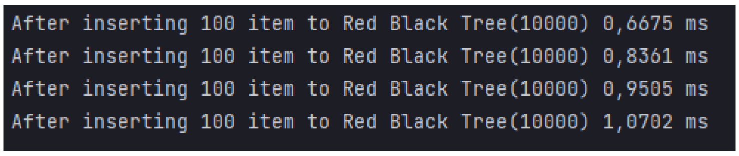
```java
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.nanoTime();
    while (count < 100) {
        int num = rand.nextInt();
        if(bSearchTrees10.get(i).add(num))
            ++count;
    }
    double end = System.nanoTime();
    totalTime10 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bSearchTrees20.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime20 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bSearchTrees40.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime40 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bSearchTrees80.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime80 += (end - start) / Math.pow(10,6);
    count = 0;
}
System.out.println("\n\n\t\t## Testing to Insert 100 item ##\n ");
System.out.printf("\nAfter inserting 100 item to Binary Search Tree(10000) %.4f ms\n",totalTime10);
System.out.printf("After inserting 100 item to Binary Search Tree(20000) %.4f ms\n",totalTime20);
System.out.printf("After inserting 100 item to Binary Search Tree(40000) %.4f ms\n",totalTime40);
System.out.printf("After inserting 100 item to Binary Search Tree(80000) %.4f ms\n",totalTime80);
```



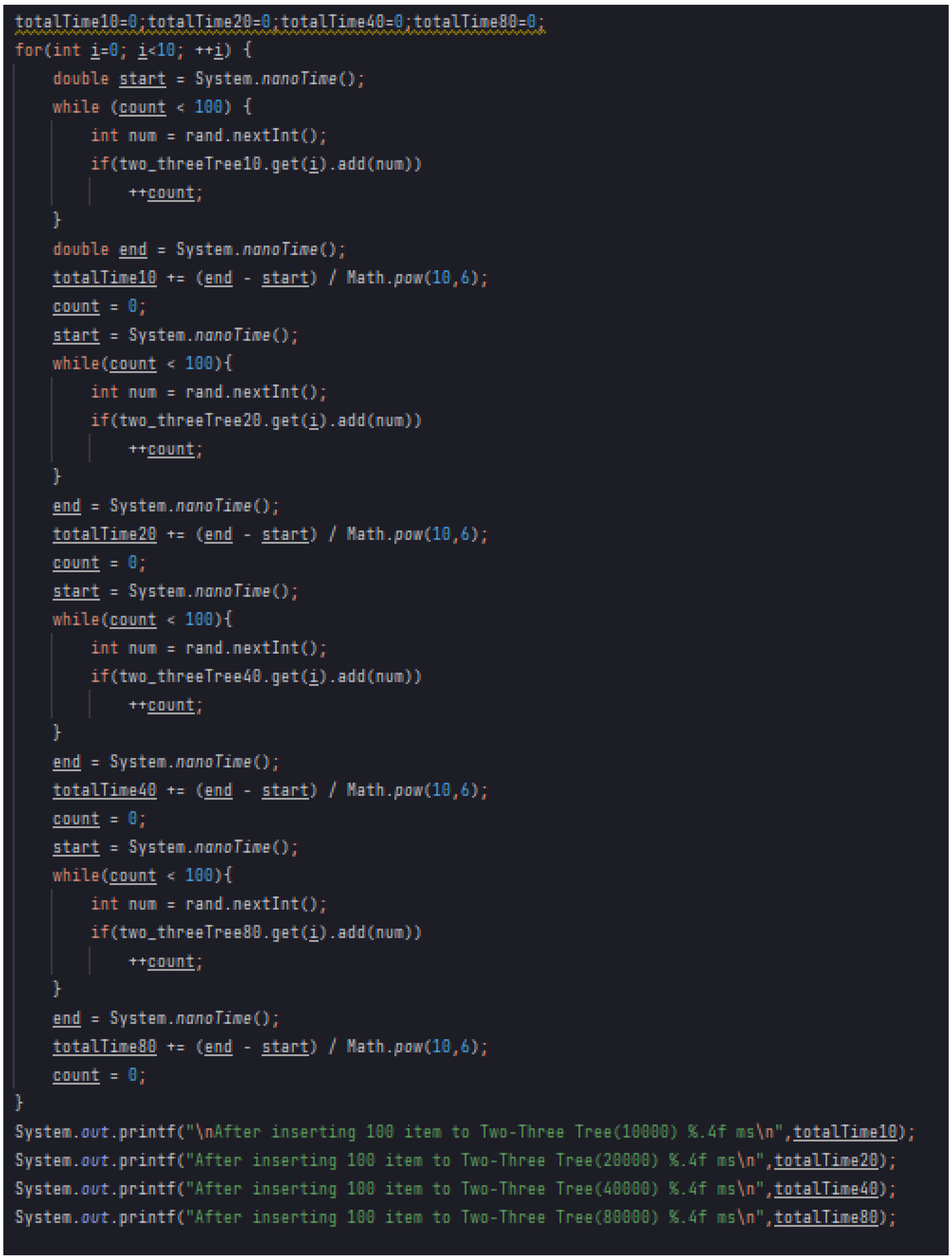
```
After inserting 100 item to Binary Search Tree(10000) 1,0577 ms
After inserting 100 item to Binary Search Tree(10000) 1,1819 ms
After inserting 100 item to Binary Search Tree(10000) 1,3911 ms
After inserting 100 item to Binary Search Tree(10000) 1,4643 ms
```



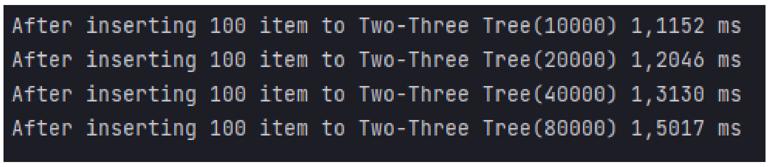
```
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.nanoTime();
    while (count < 100) {
        int num = rand.nextInt();
        if(redBlackTreesTree10.get(i).add(num))
            ++count;
    }
    double end = System.nanoTime();
    totalTime10 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(redBlackTreesTree20.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime20 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(redBlackTreesTree40.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime40 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(redBlackTreesTree80.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime80 += (end - start) / Math.pow(10,6);
    count = 0;
}
System.out.printf("\nAfter inserting 100 item to Red Black Tree(10000) %.4f ms\n",totalTime10);
System.out.printf("After inserting 100 item to Red Black Tree(20000) %.4f ms\n",totalTime20);
System.out.printf("After inserting 100 item to Red Black Tree(40000) %.4f ms\n",totalTime40);
System.out.printf("After inserting 100 item to Red Black Tree(80000) %.4f ms\n",totalTime80);
```



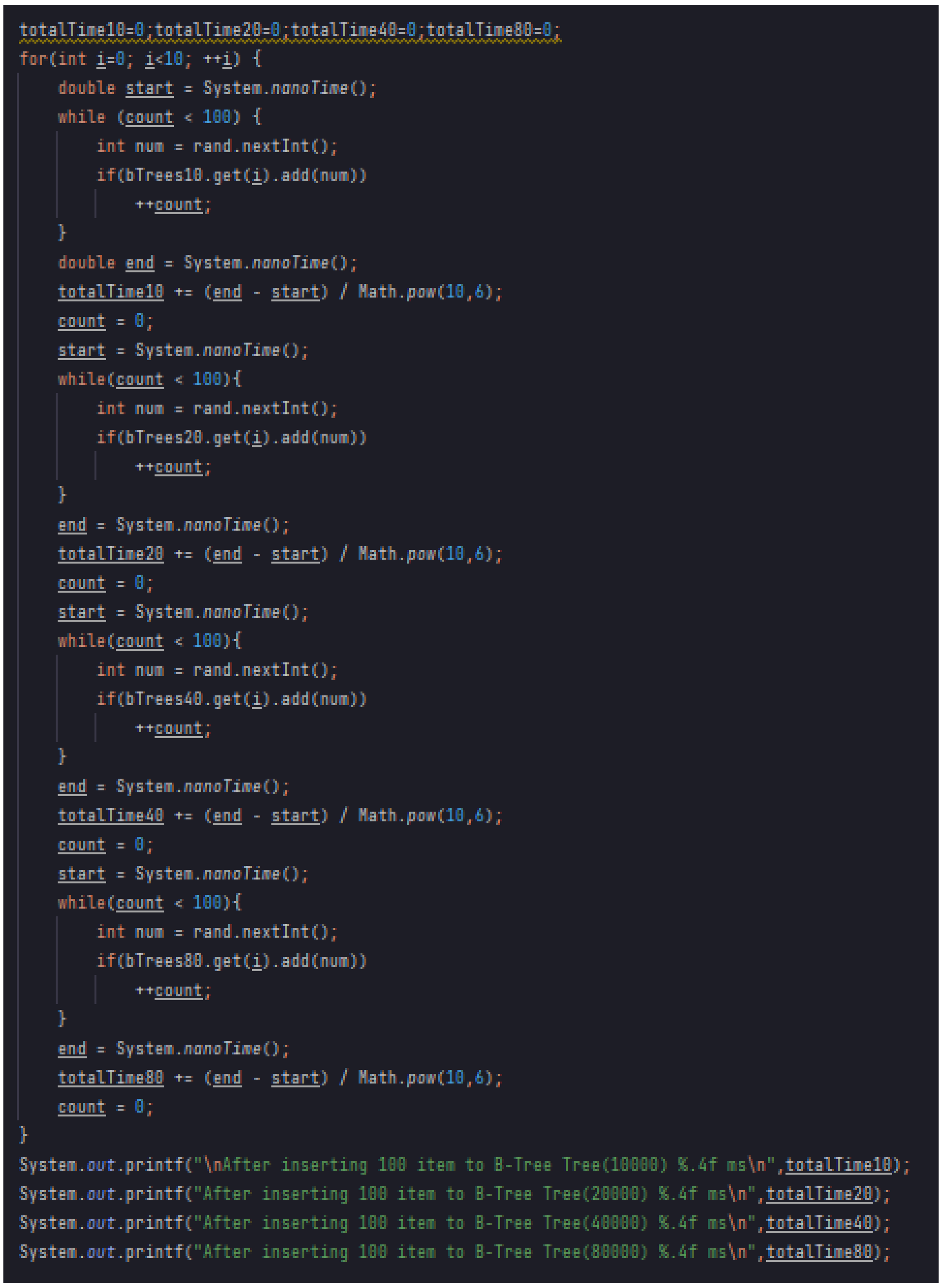
```
After inserting 100 item to Red Black Tree(10000) 0,6675 ms
After inserting 100 item to Red Black Tree(10000) 0,8361 ms
After inserting 100 item to Red Black Tree(10000) 0,9505 ms
After inserting 100 item to Red Black Tree(10000) 1,0702 ms
```



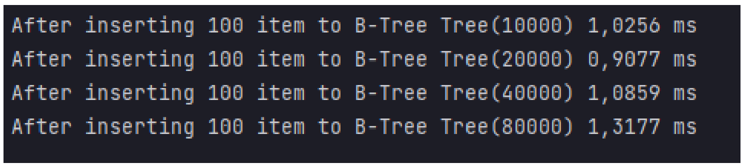
```java
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.nanoTime();
    while (count < 100) {
        int num = rand.nextInt();
        if(two_threeTree10.get(i).add(num))
            ++count;
    }
    double end = System.nanoTime();
    totalTime10 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(two_threeTree20.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime20 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(two_threeTree40.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime40 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(two_threeTree80.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime80 += (end - start) / Math.pow(10,6);
    count = 0;
}
System.out.printf("\nAfter inserting 100 item to Two-Three Tree(10000) %.4f ms\n",totalTime10);
System.out.printf("After inserting 100 item to Two-Three Tree(20000) %.4f ms\n",totalTime20);
System.out.printf("After inserting 100 item to Two-Three Tree(40000) %.4f ms\n",totalTime40);
System.out.printf("After inserting 100 item to Two-Three Tree(80000) %.4f ms\n",totalTime80);
```



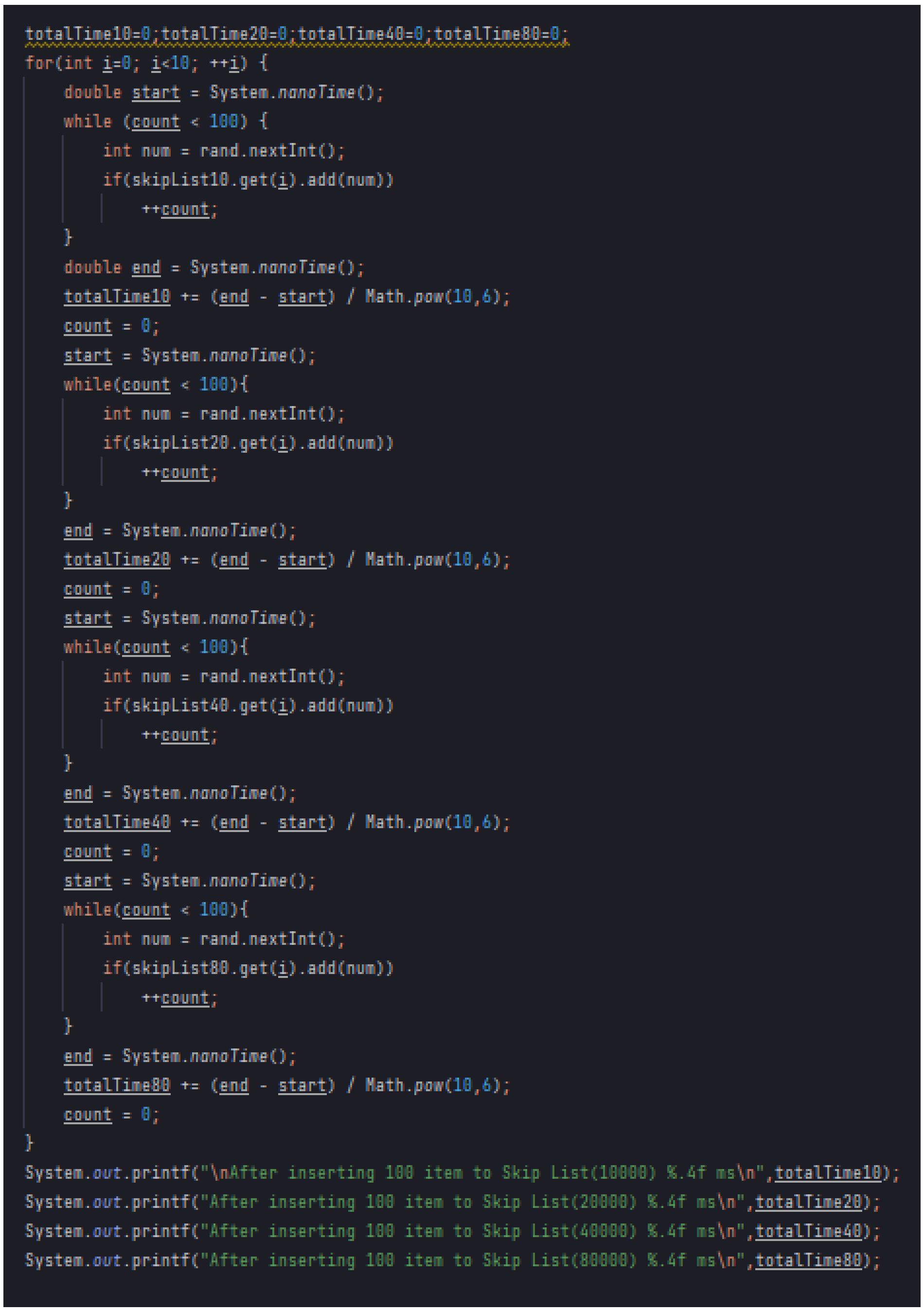
```
After inserting 100 item to Two-Three Tree(10000) 1,1152 ms
After inserting 100 item to Two-Three Tree(20000) 1,2046 ms
After inserting 100 item to Two-Three Tree(40000) 1,3130 ms
After inserting 100 item to Two-Three Tree(80000) 1,5017 ms
```



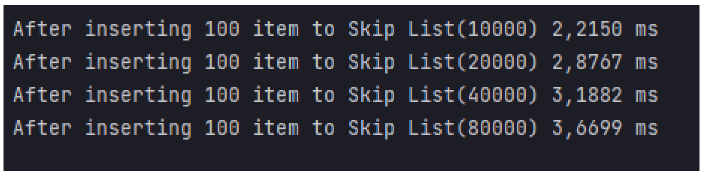
```java
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.nanoTime();
    while (count < 100) {
        int num = rand.nextInt();
        if(bTrees10.get(i).add(num))
            ++count;
    }
    double end = System.nanoTime();
    totalTime10 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bTrees20.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime20 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bTrees40.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime40 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(bTrees80.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime80 += (end - start) / Math.pow(10,6);
    count = 0;
}
System.out.printf("\nAfter inserting 100 item to B-Tree Tree(10000) %.4f ms\n",totalTime10);
System.out.printf("After inserting 100 item to B-Tree Tree(20000) %.4f ms\n",totalTime20);
System.out.printf("After inserting 100 item to B-Tree Tree(40000) %.4f ms\n",totalTime40);
System.out.printf("After inserting 100 item to B-Tree Tree(80000) %.4f ms\n",totalTime80);
```



```
After inserting 100 item to B-Tree Tree(10000) 1,0256 ms
After inserting 100 item to B-Tree Tree(20000) 0,9077 ms
After inserting 100 item to B-Tree Tree(40000) 1,0859 ms
After inserting 100 item to B-Tree Tree(80000) 1,3177 ms
```

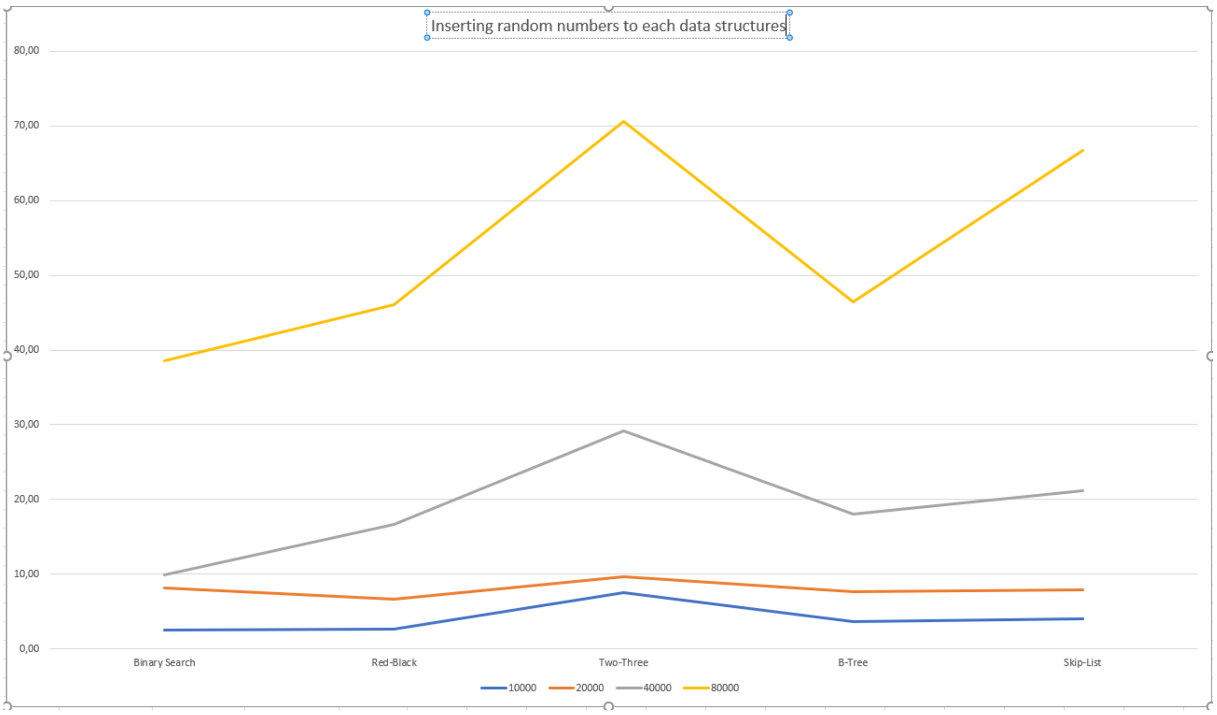


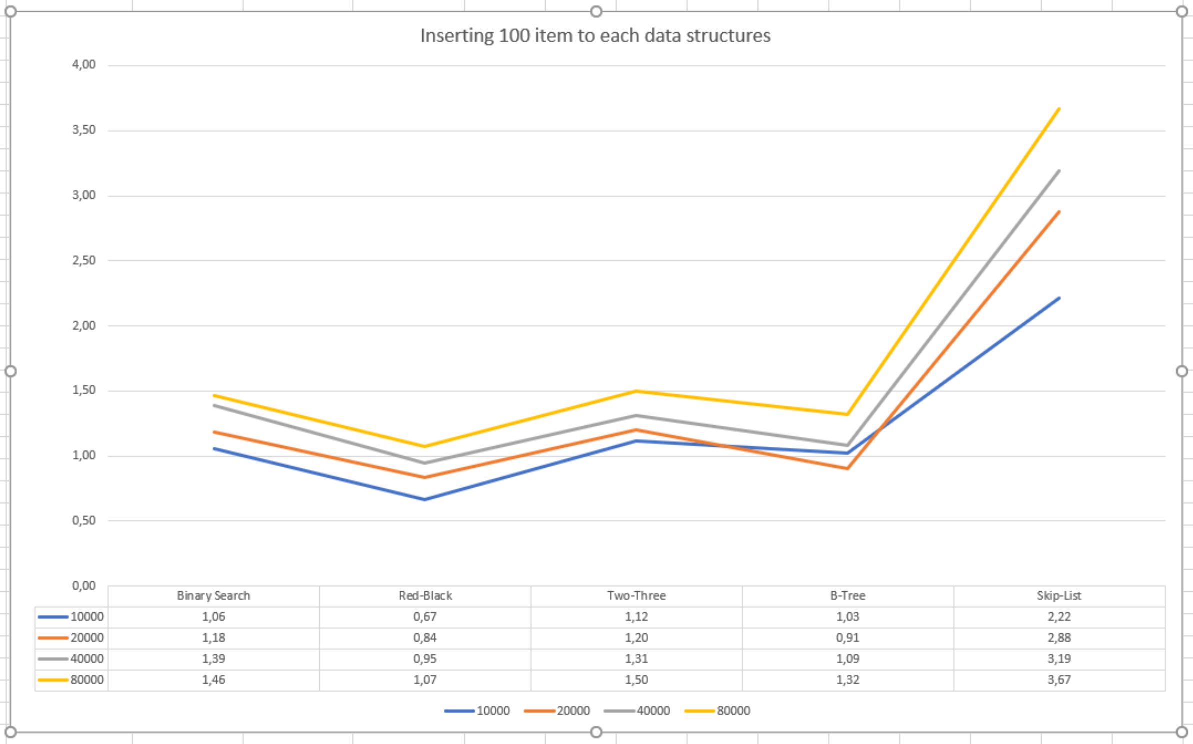
```java
totalTime10=0;totalTime20=0;totalTime40=0;totalTime80=0;
for(int i=0; i<10; ++i) {
    double start = System.nanoTime();
    while (count < 100) {
        int num = rand.nextInt();
        if(skipList10.get(i).add(num))
            ++count;
    }
    double end = System.nanoTime();
    totalTime10 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(skipList20.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime20 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(skipList40.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime40 += (end - start) / Math.pow(10,6);
    count = 0;
    start = System.nanoTime();
    while(count < 100){
        int num = rand.nextInt();
        if(skipList80.get(i).add(num))
            ++count;
    }
    end = System.nanoTime();
    totalTime80 += (end - start) / Math.pow(10,6);
    count = 0;
}
System.out.printf("\nAfter inserting 100 item to Skip List(10000) %.4f ms\n",totalTime10);
System.out.printf("After inserting 100 item to Skip List(20000) %.4f ms\n",totalTime20);
System.out.printf("After inserting 100 item to Skip List(40000) %.4f ms\n",totalTime40);
System.out.printf("After inserting 100 item to Skip List(80000) %.4f ms\n",totalTime80);
```



```
After inserting 100 item to Skip List(10000) 2,2150 ms
After inserting 100 item to Skip List(20000) 2,8767 ms
After inserting 100 item to Skip List(40000) 3,1882 ms
After inserting 100 item to Skip List(80000) 3,6699 ms
```

Running Times Graphs





	Binary Search	Red-Black	Two-Three	B-Tree	Skip-List
10000	1,06	0,67	1,12	1,03	2,22
20000	1,18	0,84	1,20	0,91	2,88
40000	1,39	0,95	1,31	1,09	3,19
80000	1,46	1,07	1,50	1,32	3,67