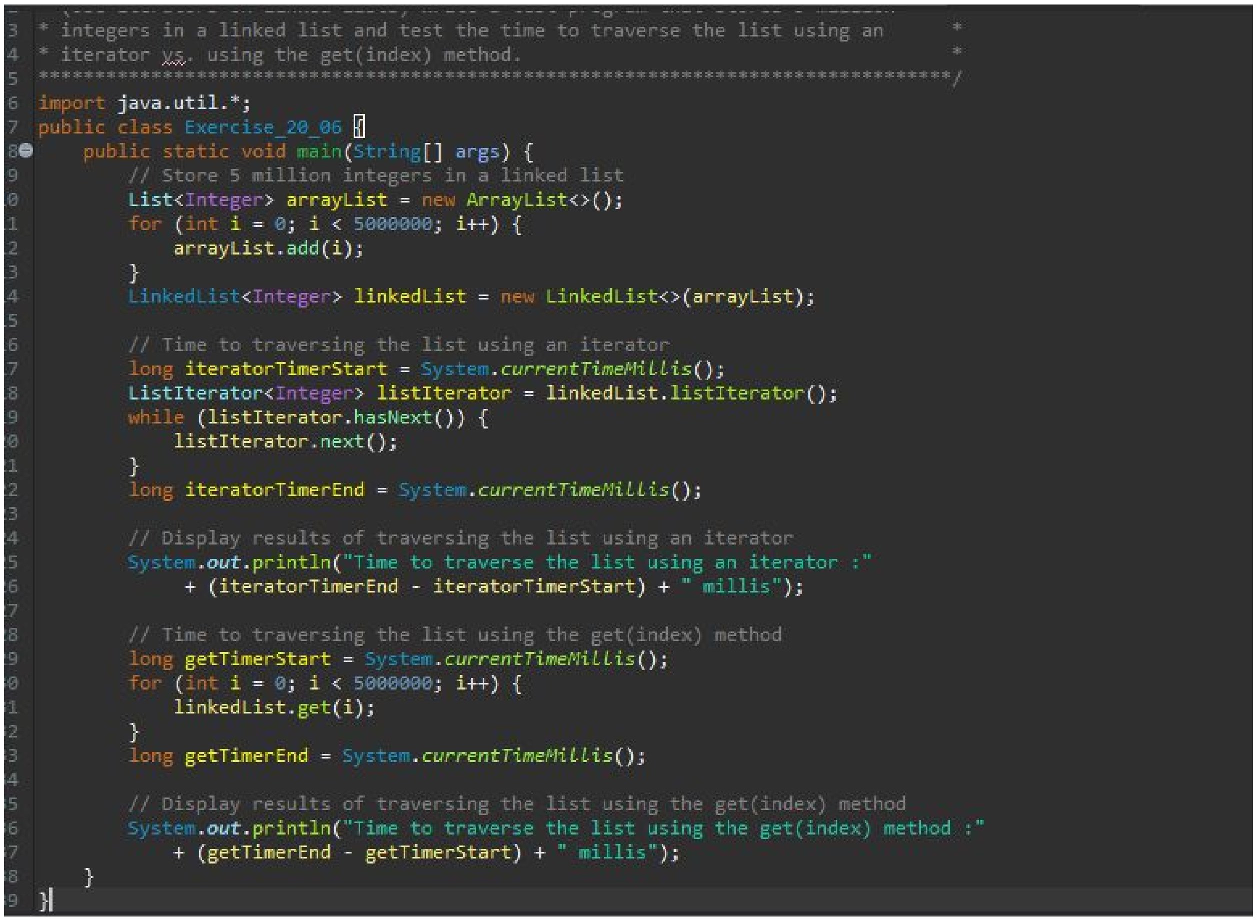
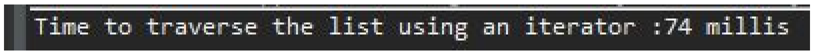
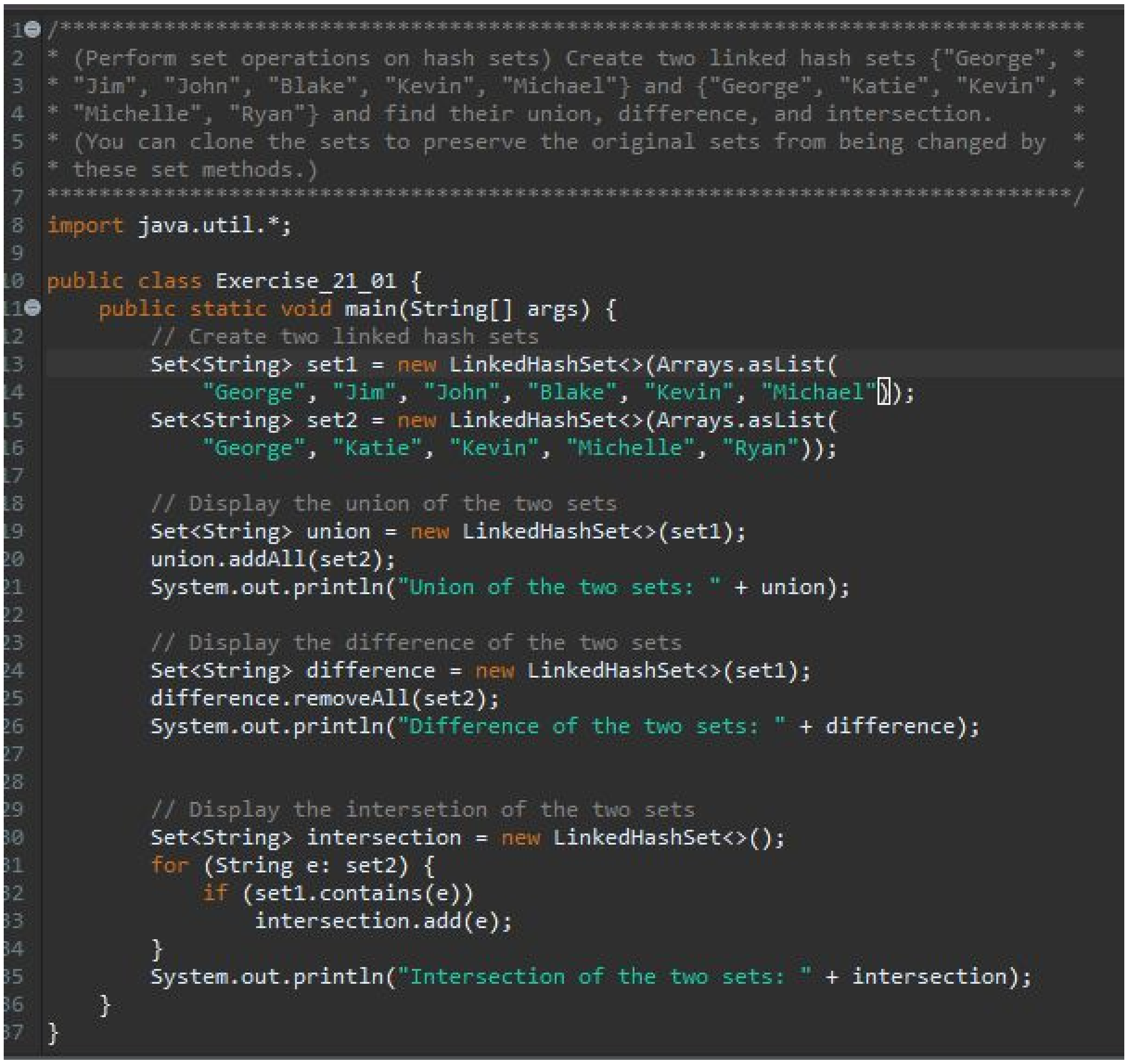
Brandon London 2261 project 5 5/5/2019



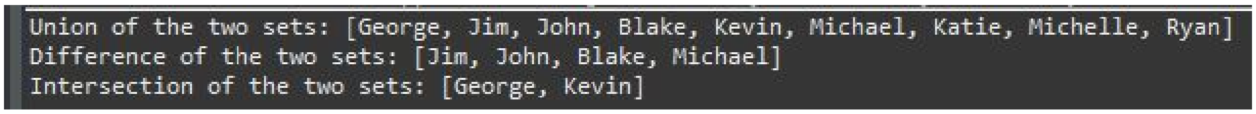
```java
 * integers in a linked list and test the time to traverse the list using an      *
 * iterator vs. using the get(index) method.                                       *
 **********************************************************************************/
import java.util.*;
public class Exercise_20_06 {
    public static void main(String[] args) {
        // Store 5 million integers in a linked list
        List<Integer> arrayList = new ArrayList<>();
        for (int i = 0; i < 5000000; i++) {
            arrayList.add(i);
        }
        LinkedList<Integer> linkedList = new LinkedList<>(arrayList);

        // Time to traversing the list using an iterator
        long iteratorTimerStart = System.currentTimeMillis();
        ListIterator<Integer> listIterator = linkedList.listIterator();
        while (listIterator.hasNext()) {
            listIterator.next();
        }
        long iteratorTimerEnd = System.currentTimeMillis();

        // Display results of traversing the list using an iterator
        System.out.println("Time to traverse the list using an iterator :"
             + (iteratorTimerEnd - iteratorTimerStart) + " millis");

        // Time to traversing the list using the get(index) method
        long getTimerStart = System.currentTimeMillis();
        for (int i = 0; i < 5000000; i++) {
            linkedList.get(i);
        }
        long getTimerEnd = System.currentTimeMillis();

        // Display results of traversing the list using the get(index) method
        System.out.println("Time to traverse the list using the get(index) method :"
            + (getTimerEnd - getTimerStart) + " millis");
    }
}
```



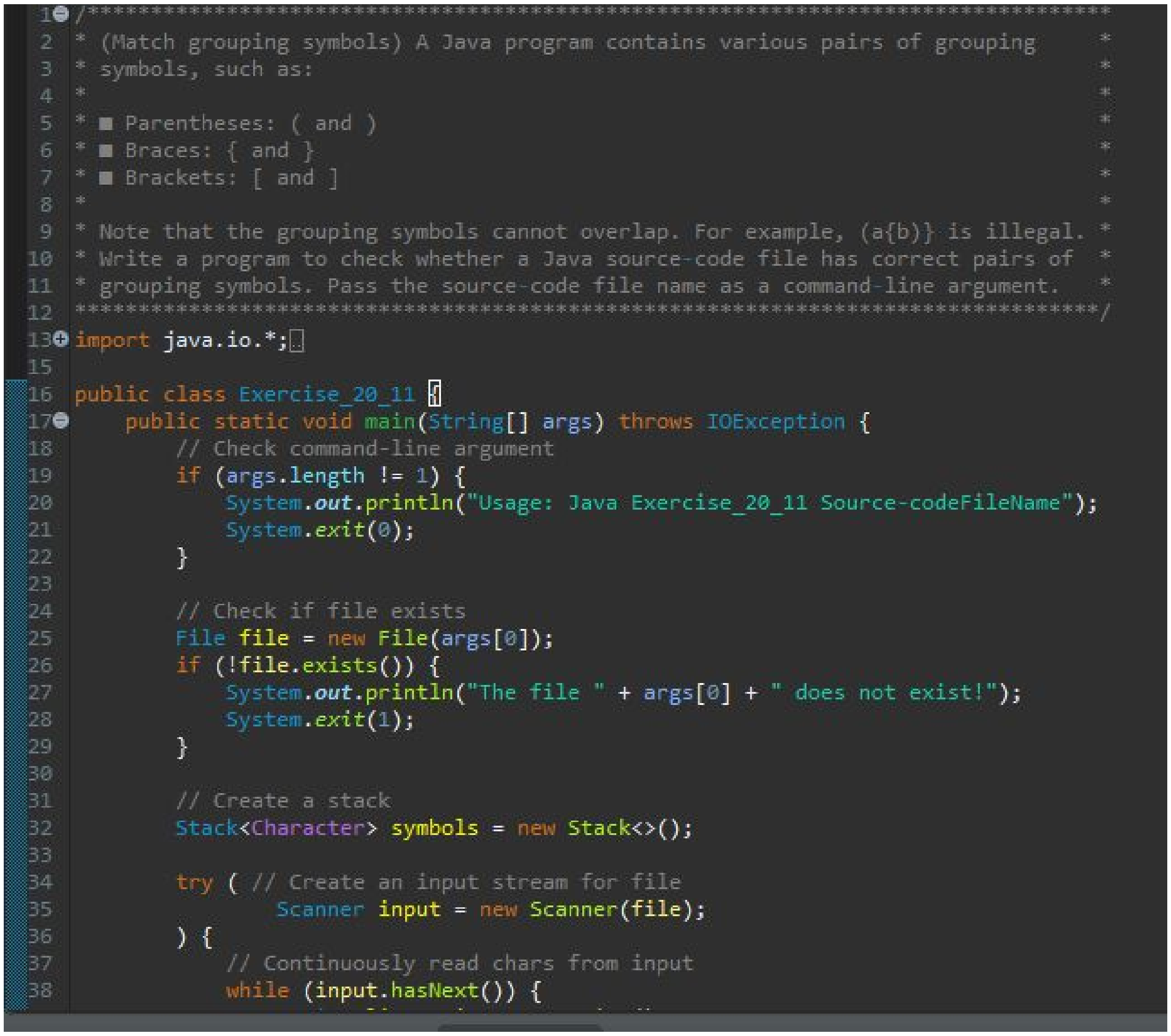
```
Time to traverse the list using an iterator :74 millis
```



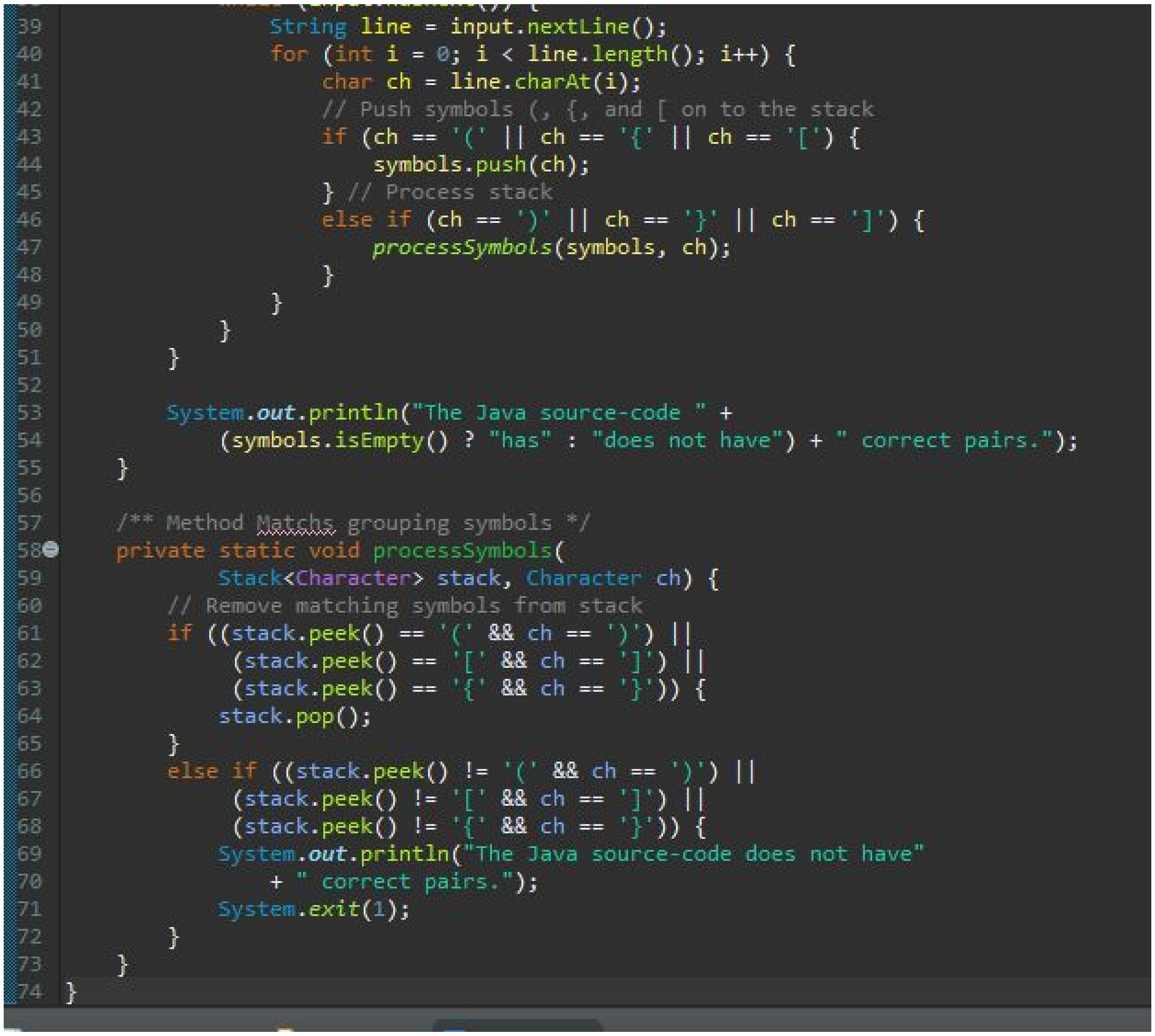
```java
/*********************************************************************************
* (Perform set operations on hash sets) Create two linked hash sets {"George", *
* "Jim", "John", "Blake", "Kevin", "Michael"} and {"George", "Katie", "Kevin", *
* "Michelle", "Ryan"} and find their union, difference, and intersection.      *
* (You can clone the sets to preserve the original sets from being changed by  *
* these set methods.)                                                          *
*********************************************************************************/
import java.util.*;

public class Exercise_21_01 {
    public static void main(String[] args) {
        // Create two linked hash sets
        Set<String> set1 = new LinkedHashSet<>(Arrays.asList(
            "George", "Jim", "John", "Blake", "Kevin", "Michael"));
        Set<String> set2 = new LinkedHashSet<>(Arrays.asList(
            "George", "Katie", "Kevin", "Michelle", "Ryan"));

        // Display the union of the two sets
        Set<String> union = new LinkedHashSet<>(set1);
        union.addAll(set2);
        System.out.println("Union of the two sets: " + union);

        // Display the difference of the two sets
        Set<String> difference = new LinkedHashSet<>(set1);
        difference.removeAll(set2);
        System.out.println("Difference of the two sets: " + difference);


        // Display the intersetion of the two sets
        Set<String> intersection = new LinkedHashSet<>();
        for (String e: set2) {
            if (set1.contains(e))
                intersection.add(e);
        }
        System.out.println("Intersection of the two sets: " + intersection);
    }
}
```



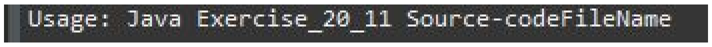
```
Union of the two sets: [George, Jim, John, Blake, Kevin, Michael, Katie, Michelle, Ryan]
Difference of the two sets: [Jim, John, Blake, Michael]
Intersection of the two sets: [George, Kevin]
```



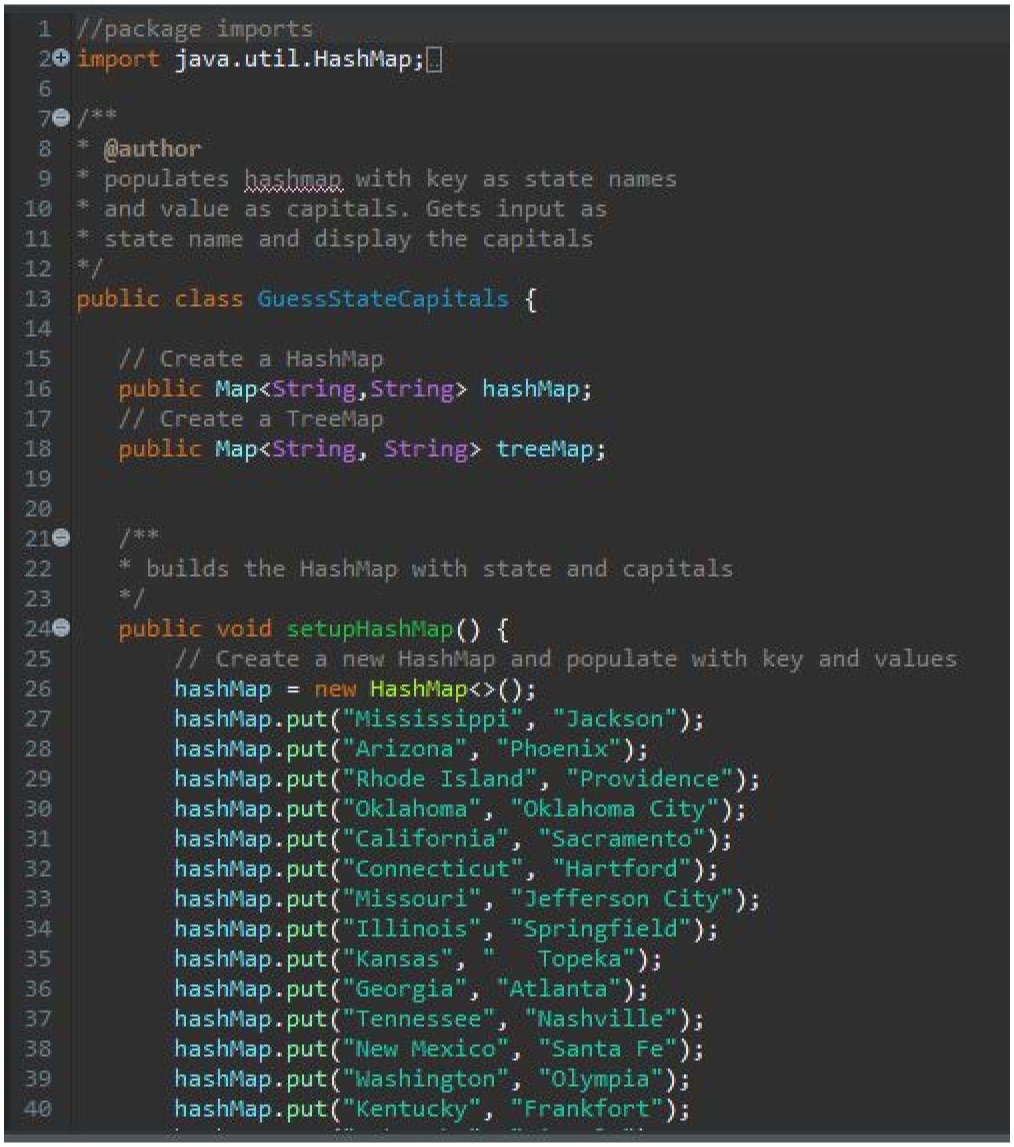
```java
/*****************************************************************************
* (Match grouping symbols) A Java program contains various pairs of grouping    *
* symbols, such as:                                                              *
*                                                                                *
* ■ Parentheses: ( and )                                                         *
* ■ Braces: { and }                                                              *
* ■ Brackets: [ and ]                                                            *
*                                                                                *
* Note that the grouping symbols cannot overlap. For example, (a{b)} is illegal. *
* Write a program to check whether a Java source-code file has correct pairs of  *
* grouping symbols. Pass the source-code file name as a command-line argument.   *
*****************************************************************************/
import java.io.*;

public class Exercise_20_11 {
    public static void main(String[] args) throws IOException {
        // Check command-line argument
        if (args.length != 1) {
            System.out.println("Usage: Java Exercise_20_11 Source-codeFileName");
            System.exit(0);
        }

        // Check if file exists
        File file = new File(args[0]);
        if (!file.exists()) {
            System.out.println("The file " + args[0] + " does not exist!");
            System.exit(1);
        }

        // Create a stack
        Stack<Character> symbols = new Stack<>();

        try ( // Create an input stream for file
                Scanner input = new Scanner(file);
        ) {
            // Continuously read chars from input
            while (input.hasNext()) {
```



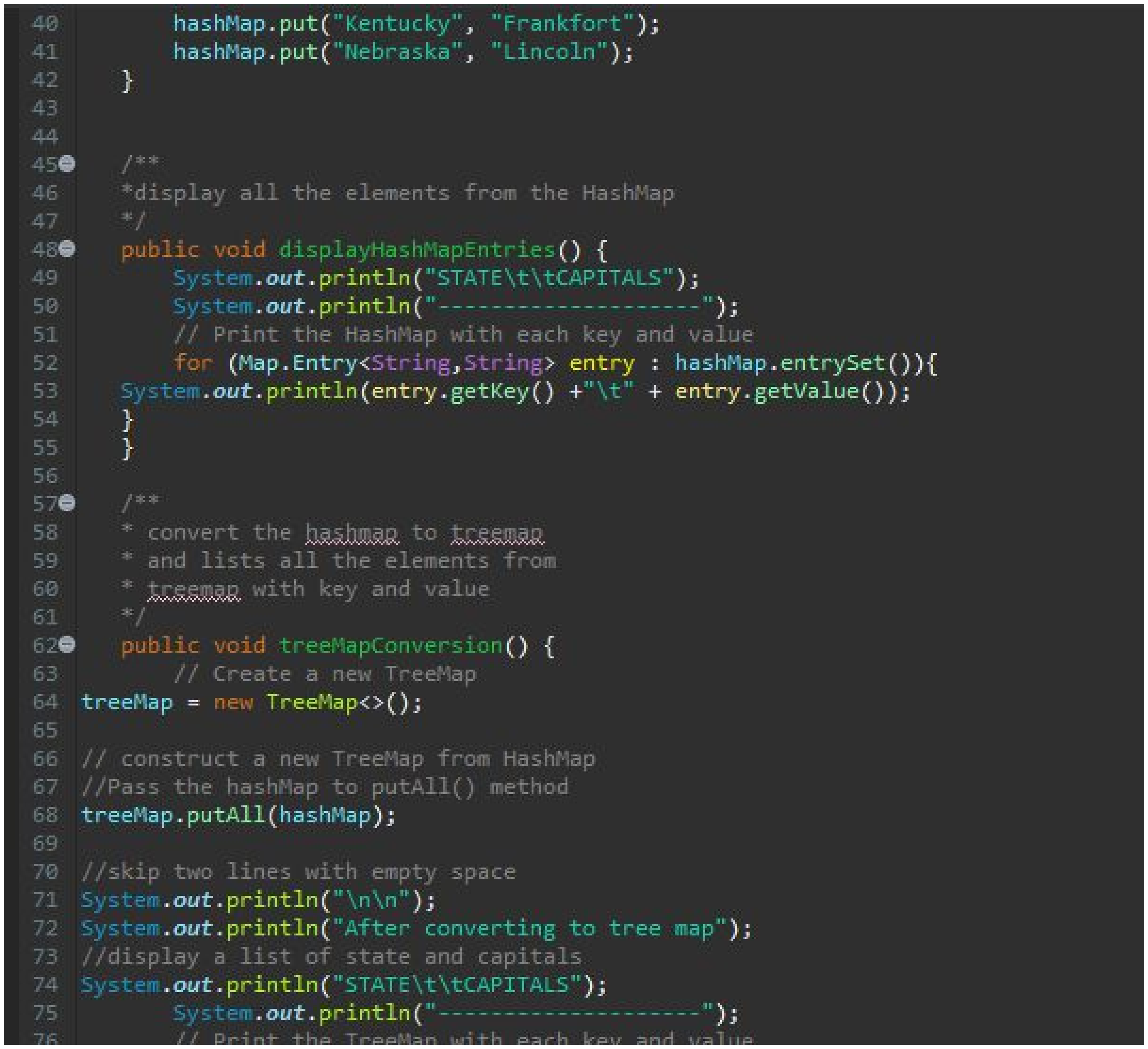
```java
                String line = input.nextLine();
                for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    // Push symbols (, {, and [ on to the stack
                    if (ch == '(' || ch == '{' || ch == '[') {
                        symbols.push(ch);
                    } // Process stack
                    else if (ch == ')' || ch == '}' || ch == ']') {
                        processSymbols(symbols, ch);
                    }
                }
            }
        }

        System.out.println("The Java source-code " +
            (symbols.isEmpty() ? "has" : "does not have") + " correct pairs.");
    }

    /** Method Matchs grouping symbols */
    private static void processSymbols(
            Stack<Character> stack, Character ch) {
        // Remove matching symbols from stack
        if ((stack.peek() == '(' && ch == ')') ||
             (stack.peek() == '[' && ch == ']') ||
             (stack.peek() == '{' && ch == '}')) {
            stack.pop();
        }
        else if ((stack.peek() != '(' && ch == ')') ||
             (stack.peek() != '[' && ch == ']') ||
             (stack.peek() != '{' && ch == '}')) {
            System.out.println("The Java source-code does not have"
                + " correct pairs.");
            System.exit(1);
        }
    }
}
```



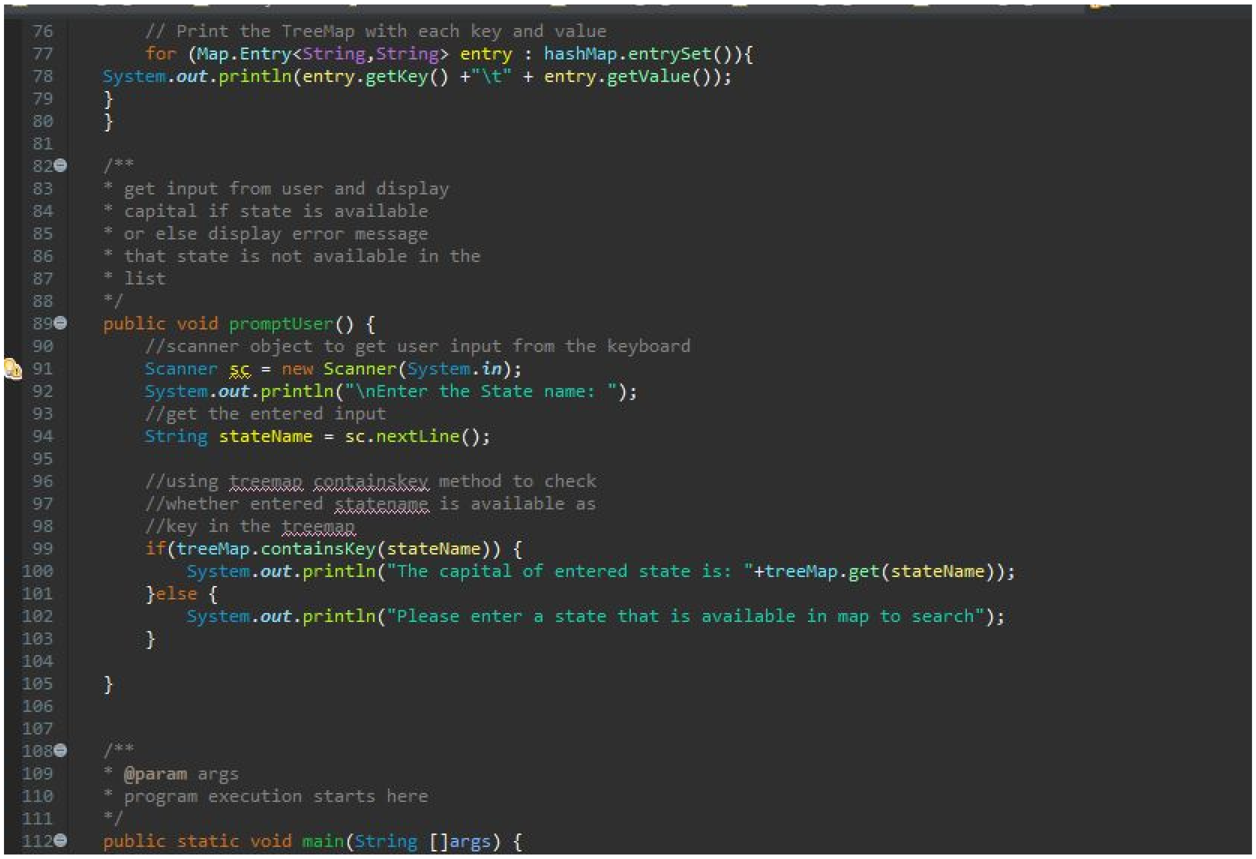
```
Usage: Java Exercise_20_11 Source-codeFileName
```



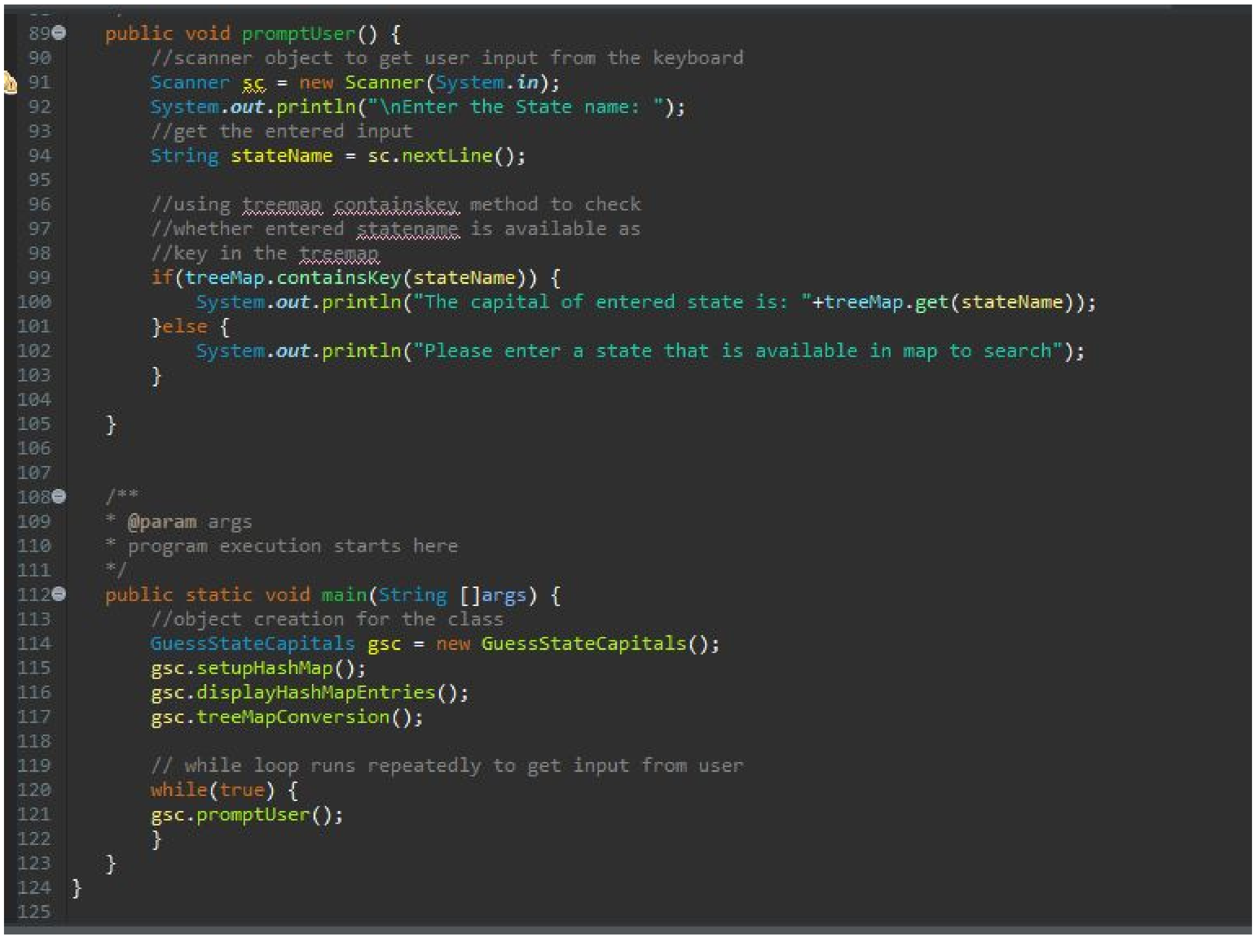
```java
//package imports
import java.util.HashMap;

/**
* @author
* populates hashmap with key as state names
* and value as capitals. Gets input as
* state name and display the capitals
*/
public class GuessStateCapitals {

    // Create a HashMap
    public Map<String,String> hashMap;
    // Create a TreeMap
    public Map<String, String> treeMap;


    /**
    * builds the HashMap with state and capitals
    */
    public void setupHashMap() {
        // Create a new HashMap and populate with key and values
        hashMap = new HashMap<>();
        hashMap.put("Mississippi", "Jackson");
        hashMap.put("Arizona", "Phoenix");
        hashMap.put("Rhode Island", "Providence");
        hashMap.put("Oklahoma", "Oklahoma City");
        hashMap.put("California", "Sacramento");
        hashMap.put("Connecticut", "Hartford");
        hashMap.put("Missouri", "Jefferson City");
        hashMap.put("Illinois", "Springfield");
        hashMap.put("Kansas", "   Topeka");
        hashMap.put("Georgia", "Atlanta");
        hashMap.put("Tennessee", "Nashville");
        hashMap.put("New Mexico", "Santa Fe");
        hashMap.put("Washington", "Olympia");
        hashMap.put("Kentucky", "Frankfort");
```



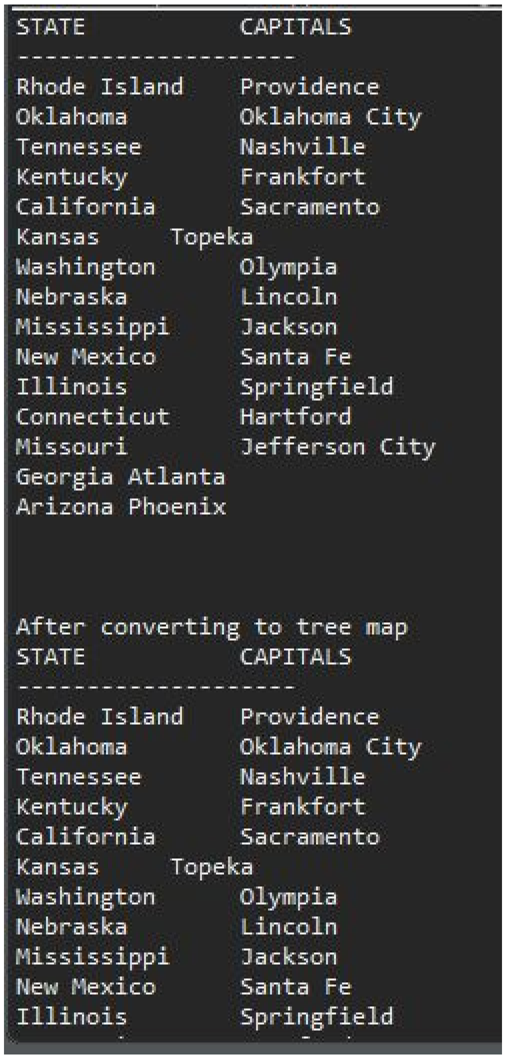
```java
        hashMap.put("Kentucky", "Frankfort");
        hashMap.put("Nebraska", "Lincoln");
    }


    /**
    *display all the elements from the HashMap
    */
    public void displayHashMapEntries() {
        System.out.println("STATE\t\tCAPITALS");
        System.out.println("--------------------");
        // Print the HashMap with each key and value
        for (Map.Entry<String,String> entry : hashMap.entrySet()){
    System.out.println(entry.getKey() +"\t" + entry.getValue());
    }
    }

    /**
    * convert the hashmap to treemap
    * and lists all the elements from
    * treemap with key and value
    */
    public void treeMapConversion() {
        // Create a new TreeMap
treeMap = new TreeMap<>();

// construct a new TreeMap from HashMap
//Pass the hashMap to putAll() method
treeMap.putAll(hashMap);

//skip two lines with empty space
System.out.println("\n\n");
System.out.println("After converting to tree map");
//display a list of state and capitals
System.out.println("STATE\t\tCAPITALS");
        System.out.println("--------------------");
        // Print the TreeMap with each key and value
```



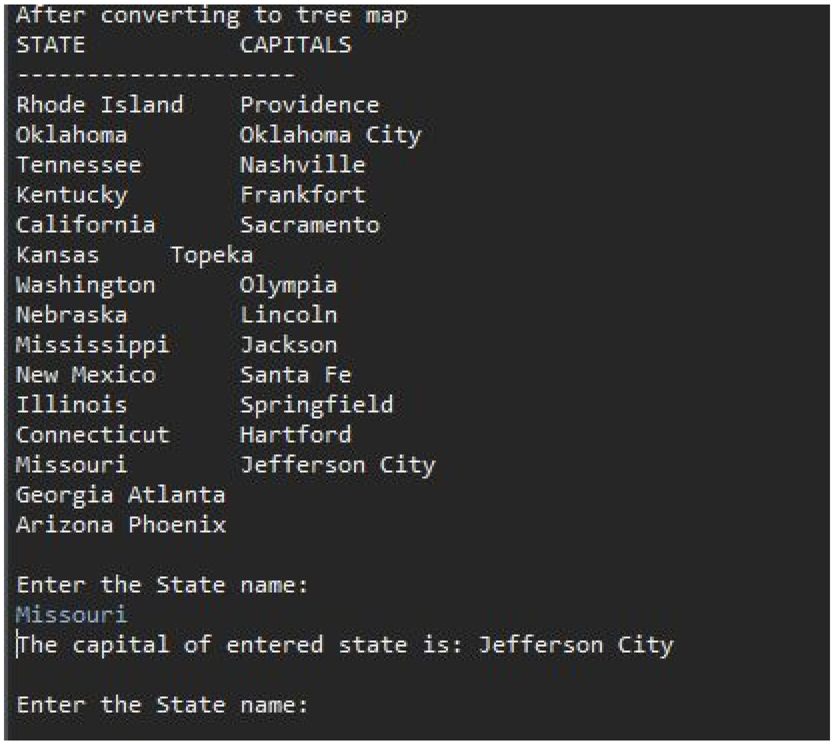
```
        // Print the TreeMap with each key and value
        for (Map.Entry<String,String> entry : hashMap.entrySet()){
    System.out.println(entry.getKey() +"\t" + entry.getValue());
    }
    }

    /**
    * get input from user and display
    * capital if state is available
    * or else display error message
    * that state is not available in the
    * list
    */
    public void promptUser() {
        //scanner object to get user input from the keyboard
        Scanner sc = new Scanner(System.in);
        System.out.println("\nEnter the State name: ");
        //get the entered input
        String stateName = sc.nextLine();

        //using treemap containskey method to check
        //whether entered statename is available as
        //key in the treemap
        if(treeMap.containsKey(stateName)) {
            System.out.println("The capital of entered state is: "+treeMap.get(stateName));
        }else {
            System.out.println("Please enter a state that is available in map to search");
        }

    }


    /**
    * @param args
    * program execution starts here
    */
    public static void main(String []args) {
```



```java
    public void promptUser() {
        //scanner object to get user input from the keyboard
        Scanner sc = new Scanner(System.in);
        System.out.println("\nEnter the State name: ");
        //get the entered input
        String stateName = sc.nextLine();

        //using treemap containskey method to check
        //whether entered statename is available as
        //key in the treemap
        if(treeMap.containsKey(stateName)) {
            System.out.println("The capital of entered state is: "+treeMap.get(stateName));
        }else {
            System.out.println("Please enter a state that is available in map to search");
        }

    }


    /**
    * @param args
    * program execution starts here
    */
    public static void main(String []args) {
        //object creation for the class
        GuessStateCapitals gsc = new GuessStateCapitals();
        gsc.setupHashMap();
        gsc.displayHashMapEntries();
        gsc.treeMapConversion();

        // while loop runs repeatedly to get input from user
        while(true) {
        gsc.promptUser();
        }
    }
}
```



```
STATE           CAPITALS
--------------------
Rhode Island    Providence
Oklahoma        Oklahoma City
Tennessee       Nashville
Kentucky        Frankfort
California      Sacramento
Kansas     Topeka
Washington      Olympia
Nebraska        Lincoln
Mississippi     Jackson
New Mexico      Santa Fe
Illinois        Springfield
Connecticut     Hartford
Missouri        Jefferson City
Georgia Atlanta
Arizona Phoenix



After converting to tree map
STATE           CAPITALS
--------------------
Rhode Island    Providence
Oklahoma        Oklahoma City
Tennessee       Nashville
Kentucky        Frankfort
California      Sacramento
Kansas     Topeka
Washington      Olympia
Nebraska        Lincoln
Mississippi     Jackson
New Mexico      Santa Fe
Illinois        Springfield
```



```
After converting to tree map
STATE           CAPITALS
--------------------
Rhode Island    Providence
Oklahoma        Oklahoma City
Tennessee       Nashville
Kentucky        Frankfort
California      Sacramento
Kansas     Topeka
Washington      Olympia
Nebraska        Lincoln
Mississippi     Jackson
New Mexico      Santa Fe
Illinois        Springfield
Connecticut     Hartford
Missouri        Jefferson City
Georgia Atlanta
Arizona Phoenix

Enter the State name:
Missouri
The capital of entered state is: Jefferson City

Enter the State name:
```