**Министерство науки и высшего образования Российской Федерации**  
  
Федеральное государственное автономное образовательное учреждение  
высшего образования «Уральский федеральный университет  
имени первого Президента России Б.Н. Ельцина»  
  
Институт радиоэлектроники и информационных технологий – РТФ  
Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9**по дисциплине «Курсовое проектирование Java»

**Тема: «Введение в алгоритмы и структуры данных. Исследование возможностей коллекций в языке программирования Java»**

|  |  |
| --- | --- |
| Студент группы РИВ-410938у | М.О. Покуль |
| Преподаватель: | Н.А. Архипов |

**Екатеринбург 2024**

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc190808272)

[2 Описание задачи 4](#_Toc190808273)

[3 Ход выполнения работы. 6](#_Toc190808274)

[3.1 Приложения для демонстрации примеров 1 – 5. 6](#_Toc190808275)

[3.2 Приложение с использованием рекурсии для перевода целого числа, введенного с клавиатуры, в двоичную систему счисления. 8](#_Toc190808276)

[3.3 Приложение, позволяющее ввести и вывести одномерный массив целых чисел. 9](#_Toc190808277)

[3.4 Пример 1 из раздела 2 с обработкой кода с помощью отладчика и скриншоты трех точек, обработанных отладчиком. 11](#_Toc190808278)

[3.5 Два способа создания линейного однонаправленного списка (с головы и с хвоста) согласно примеру 2 из второго раздела и отработка кода с помощью отладчика. 14](#_Toc190808279)

[3.7 Две программы, моделирующие процесс, где при ведении счета по кругу вычеркивается каждый второй человек, пока не останется один. Одна из программ использует класс ArrayList, а вторая — LinkedList. 19](#_Toc190808280)

[3.8 Проект, в котором для ввода, вывода и изменения односвязного линейного списка созданы требуемые методы. 20](#_Toc190808281)

1. Постановка задачи

Приобретение навыков работы с рекурсивными методами, знакомство с динамическими структурами данных, приобретение навыков создания и использования простейшей динамической структуры, исследование возможностей коллекций в языке программирования Java, сравнение коллекций по времени выполнения операций, знакомство с реализацией коллекций.

1. Описание задачи
2. Создать приложения для демонстрации примеров 1 – 5 из раздела 1. Для примера 5 дополнительно вывести в консоль последовательность обхода дерева рекурсивных вызовов.
3. Создать приложение с использованием рекурсии для перевода целого числа, введенного с клавиатуры, в двоичную систему счисления.
4. Создать приложение, позволяющее ввести и вывести одномерный массив целых чисел. Для ввода и вывода массива разработать рекурсивные методы вместо циклов for.
5. Выполнить пример 1 из раздела 2. Отработать код с помощью отладчика и привести скриншоты минимум трех точек, обработанных отладчиком.
6. Создать два проекта, в которых продемонстрировать два способа создания линейного однонаправленного списка (с головы и с хвоста) согласно примеру 2 из второго раздела. Отработать код с помощью отладчика и привести скриншоты минимум трех точек, обработанных отладчиком.
7. Заполнить HashMap 10 объектами <Integer, String>. Найти строки у которых ключ> 5. Если ключ = 0, вывести строки через запятую. Перемножить все ключи, где длина строки>5.
8. В кругу стоят N человек, пронумерованных от 1 до N. При ведении счета по кругу вычеркивается каждый второй человек, пока не останется один. Составить две программы, моделирующие процесс. Одна из программ должна использовать класс ArrayList, а вторая — LinkedList. Какая из двух программ работает быстрее? Почему?
9. Разработать проект, в котором для ввода, вывода и изменения односвязного линейного списка создать следующие методы:
10. с использованием цикла:

* ввод с головы createHead();
* ввод с хвоста createTail();
* вывод (возвращается строка, сформированная из элементов списка) toString();
* добавление элемента в начало списка AddFirst();
* добавление элемента в конец списка AddLast();
* вставка элемента в список с указанным номером Insert();
* удаление элемента с головы списка RemoveFirst();
* удаление последнего элемента списка RemoveLast();
* удаление из списка элемента с указанным номером
* Remove();

1. с использованием рекурсии:

* ввод с головы createHeadRec();
* ввод с хвоста createTailRec();
* вывод (возвращается строка, сформированная из элементов списка) toStringRec().

1. Ход выполнения работы.
   1. Приложения для демонстрации примеров 1 – 5.
2. Приложение 1

package Lab9.Task1;  
  
public class Recursion1 {  
 public static void m(int x){  
 System.*out*.println("x="+x);  
 if ((2\*x+1)<20){  
 *m*(2\*x+1);  
 }  
 }  
 public static void main(String[] args){  
 *m*(1);  
 }  
}

1. Приложение 2

package Lab9.Task1;  
  
public class Recursion2 {  
 public static void m(int x){  
 if ((2\*x+1)<20){  
 *m*(2\*x+1);  
 }  
 System.*out*.println("x="+x);  
 }  
 public static void main(String[] args){  
 *m*(1);  
 }  
}

1. Приложение 3

package Lab9.Task1;  
  
public class Recursion3 {  
 private static int *step* = 0;  
 public static void m(int x){  
 *space*();  
 System.*out*.println(""+x+"-> ");  
 *step*++;  
 if ((2\*x+1)<20){  
 *m*(2\*x+1);  
 }  
 *step*--;  
 *space*();  
 System.*out*.println(""+x+" <-");  
 }  
 public static void space(){  
 for (int i = 0; i < *step*; i++){  
 System.*out*.println(" ");  
 }  
 }  
 public static void main(String[] args){  
 *m*(1);  
 }  
}

1. Приложение 4

package Lab9.Task1;  
  
public class Recursion4 {  
 public static void main(String[] args) {  
 System.*out*.println(*fact*(5));  
 }  
 public static int fact(int n){  
 int result;  
 if (n == 1) return 1;  
 else {  
 result = *fact*(n-1)\*n;  
 return result;  
 }  
 }  
}

1. Приложение 5

package Lab9.Task1;  
  
public class Recursion5 {  
 public static void main(String[] args) {  
 System.*out*.println("Факториал:" + *fact*(5, 1));  
 }  
 public static int fact(int num, int layer){  
 System.*out*.println(String.*format*("%" + layer + "s>", num).replace(' ', '-'));  
 if (num==0){  
 System.*out*.println(String.*format*("%" + layer + "s<", num).replace(' ', '-'));  
 return 0;  
 } else if (num==1) {  
 System.*out*.println(String.*format*("%" + layer + "s<", num).replace(' ', '-'));  
 return 1;  
 }else {  
 int e = *fact*(num-2, layer+1)+*fact*(num-1, layer+1);  
 System.*out*.println(String.*format*("%" + layer + "s<", num).replace(' ', '-'));  
 return e;  
 }  
 }  
}

* 1. Приложение с использованием рекурсии для перевода целого числа, введенного с клавиатуры, в двоичную систему счисления.

package Lab9;  
  
import java.util.Scanner;  
public class Task2 {  
  
 // Рекурсивный метод для перевода числа в двоичную систему  
 public static String decToBin(int number) {  
 // Базовый случай: если число равно 0, возвращаем пустую строку  
 if (number == 0) {  
 return "0";  
 }  
 // Рекурсивный вызов, делим число на 2 и получаем остаток  
 return decToBin(number / 2) + (number % 2);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Введите целое число: ");  
  
 // Читаем целое число с клавиатуры  
 int number = scanner.nextInt();  
  
 // Проверяем, не является ли число отрицательным  
 if (number < 0) {  
 System.out.println("Пожалуйста, введите неотрицательное целое число.");  
 } else {  
 // Получаем двоичное представление числа  
 String binaryRepresentation = decToBin(number);  
 System.out.println("Двоичное представление числа " + number + " равно: " + binaryRepresentation);  
 }  
 }  
}

* 1. Приложение, позволяющее ввести и вывести одномерный массив целых чисел.

package Lab9;  
  
import java.util.Scanner;  
public class Task3 {  
  
 *// Рекурсивный метод для ввода элементов массива* public static void inputArray(int[] array, int index) {  
 *// Базовый случай: если индекс равен длине массива, завершаем рекурсию* if (index >= array.length) {  
 return;  
 }  
 *// Запрашиваем у пользователя ввод элемента массива* Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите элемент " + (index + 1) + ": ");  
 array[index] = scanner.nextInt();  
 *// Рекурсивный вызов для следующего индекса  
 inputArray*(array, index + 1);  
 }  
  
 *// Рекурсивный метод для вывода элементов массива* public static void printArray(int[] array, int index) {  
 *// Базовый случай: если индекс равен длине массива, завершаем рекурсию* if (index >= array.length) {  
 return;  
 }  
 *// Выводим текущий элемент массива* System.*out*.print(array[index] + " ");  
 *// Рекурсивный вызов для следующего индекса  
 printArray*(array, index + 1);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите размер массива: ");  
 int size = scanner.nextInt();  
  
 *// Создаем массив указанного размера* int[] array = new int[size];  
  
 *// Вводим элементы массива  
 inputArray*(array, 0);  
  
 *// Выводим элементы массива* System.*out*.println("Элементы массива:");  
 *printArray*(array, 0);  
 }  
}

* 1. Пример 1 из раздела 2 с обработкой кода с помощью отладчика и скриншоты трех точек, обработанных отладчиком.

package Lab9.Task4;  
  
public class Node {  
 public int value;  
 public Node next;  
  
 public Node(int value, Node next){  
 this.value = value;  
 this.next = next;  
 }  
}

package Lab9.Task4;  
  
public class Test {  
 public static void main(String[] args) {  
 Node node0 = new Node(0, null);  
 Node node1 = new Node(1, null);  
 Node node2 = new Node(2, null);  
 Node node3 = new Node(3, null);  
  
 node0.next = node1;  
 node1.next = node2;  
 node2.next = node3;  
  
 Node ref = node0;  
 while (ref != null){  
 System.*out*.print(" " + ref.value);  
 ref = ref.next;  
 }  
 }  
}

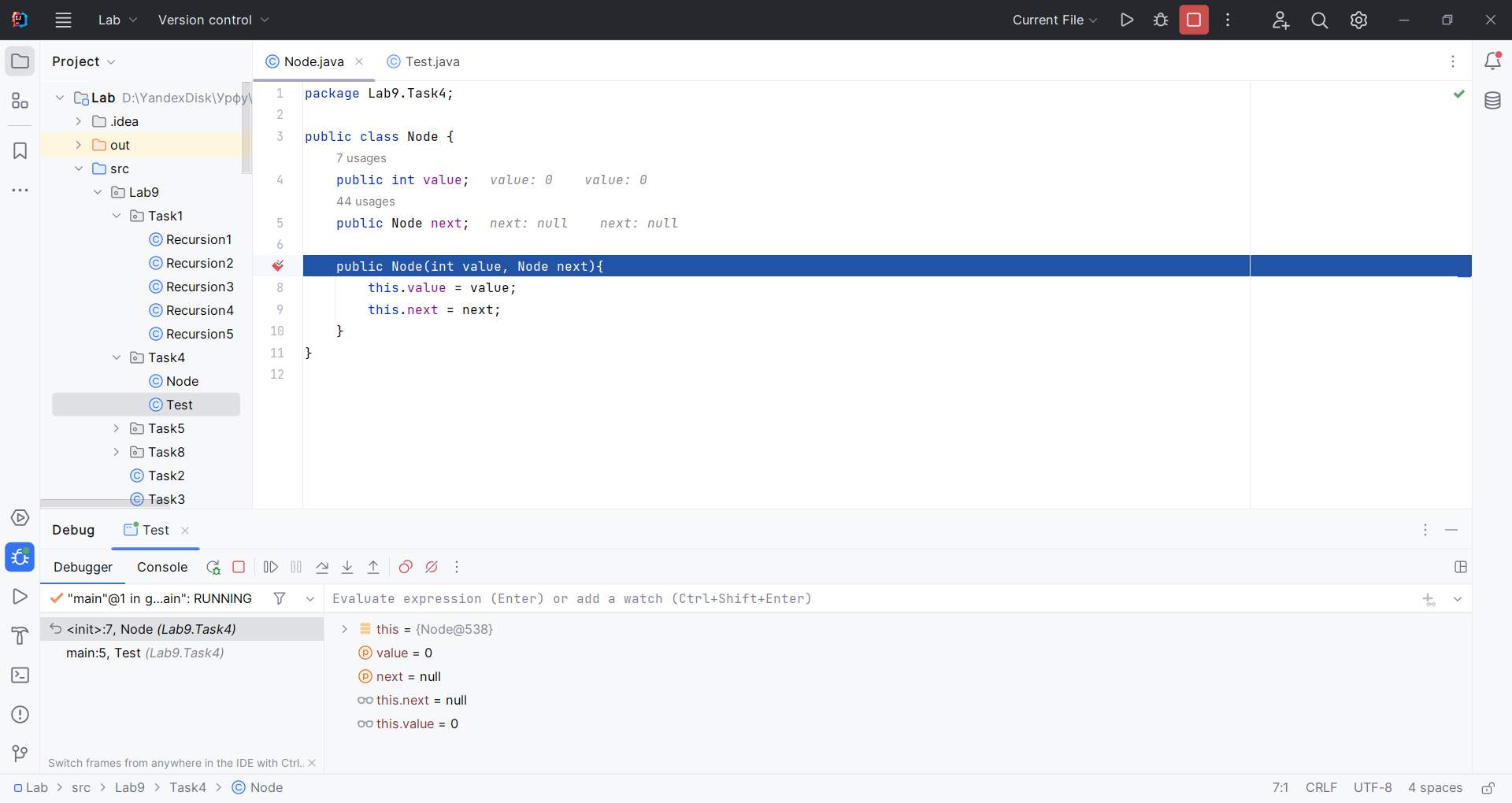


Рисунок 1 – первая точка обработки

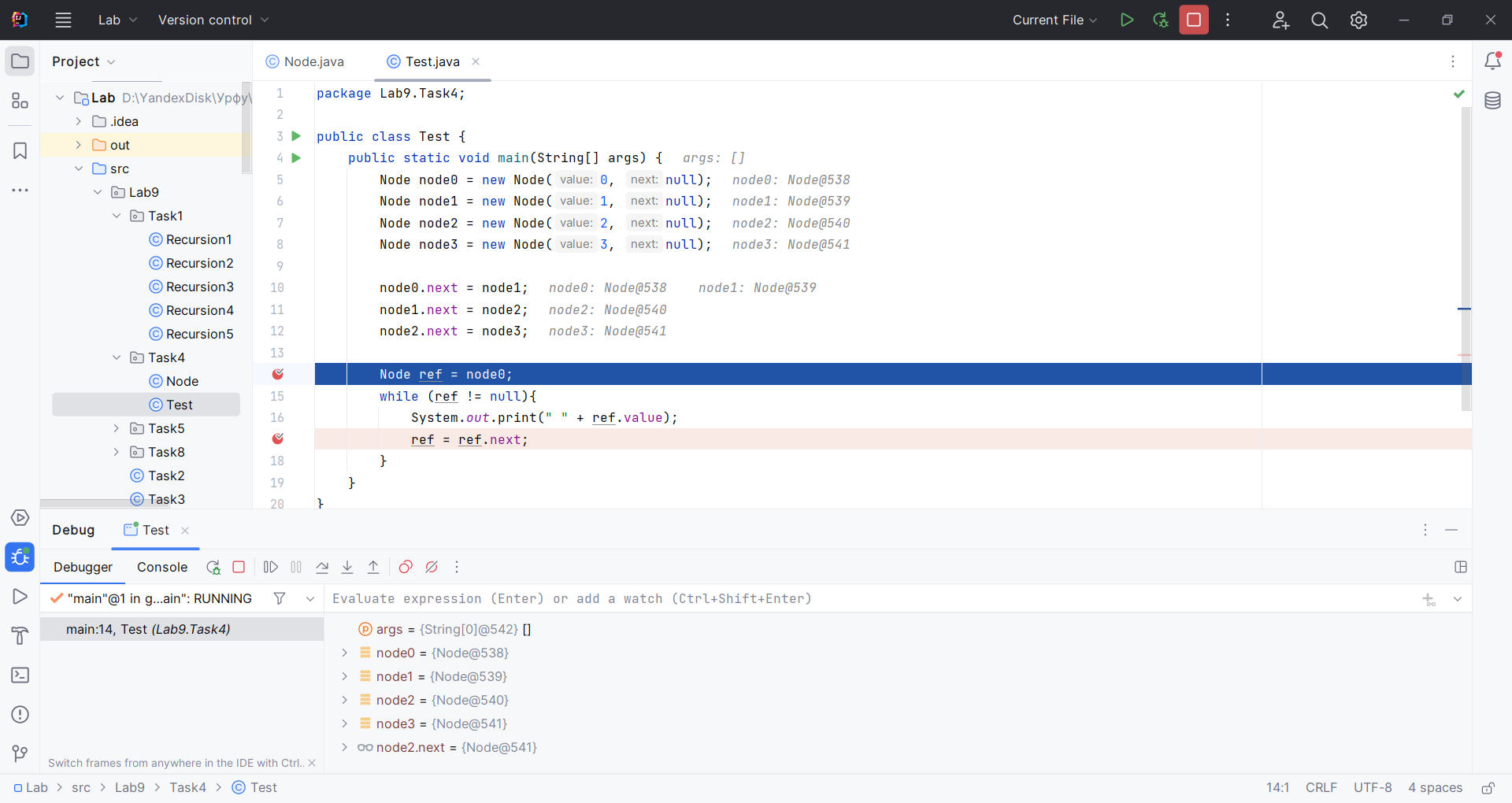


Рисунок 2 – вторая точка обработки

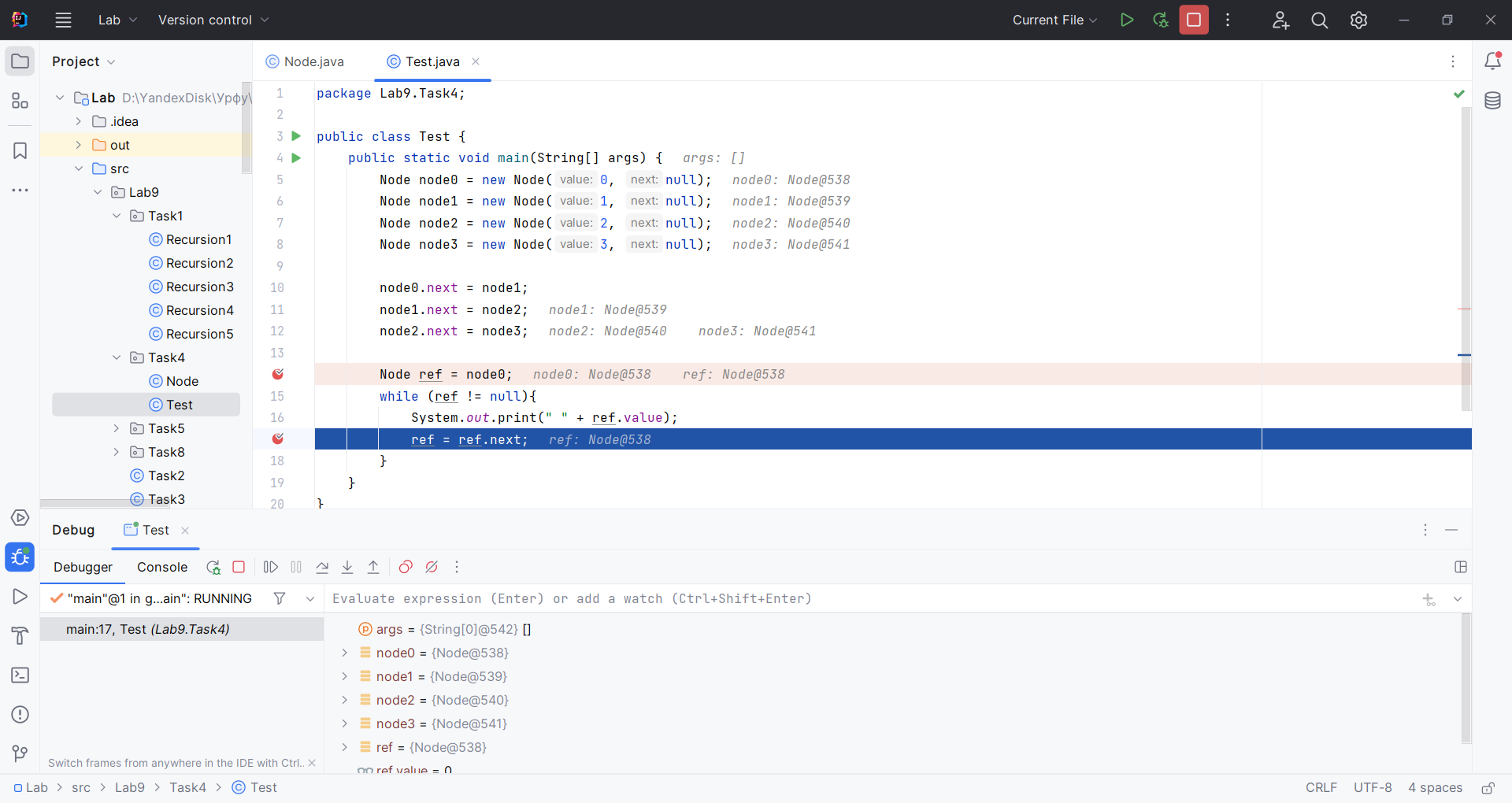


Рисунок 3 – третья точка обработки

* 1. Два способа создания линейного однонаправленного списка (с головы и с хвоста) согласно примеру 2 из второго раздела и отработка кода с помощью отладчика.

package Lab9.Task5;  
  
import Lab9.Task4.Node;  
  
public class FromHead {  
 public static void main(String[] args) {  
 Node head = null;  
 for (int i = 9; i >= 0; i--){  
 head = new Node(i, head);  
 }  
 Node ref = head;  
 while (ref != null) {  
 System.*out*.print(ref.value + " ");  
 ref = ref.next;  
 }  
 }  
}

package Lab9.Task5;  
  
import Lab9.Task4.Node;  
  
public class FromTail {  
 public static void main(String[] args) {  
 Node head = new Node(0,null);  
 for (int i = 1; i <= 9; i++){  
 Node ref = head;  
 while (ref.next != null){  
 ref = ref.next;  
 }  
 ref.next=new Node(i,null);  
 }  
 Node ref = head;  
 while (ref != null) {  
 System.*out*.print(ref.value + " ");  
 ref = ref.next;  
 }  
 }  
}

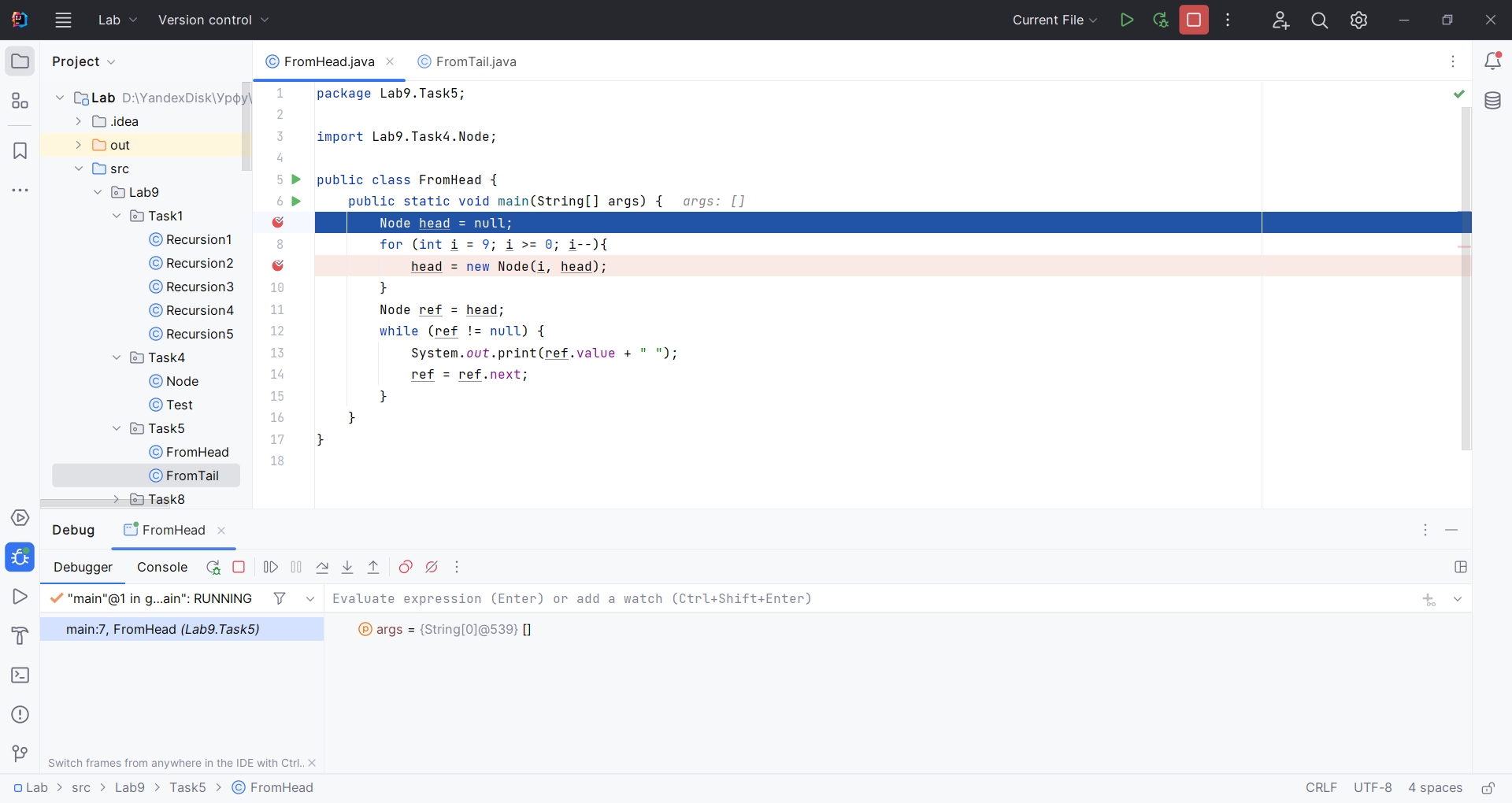


Рисунок 4 – первая точка отработки отладчика

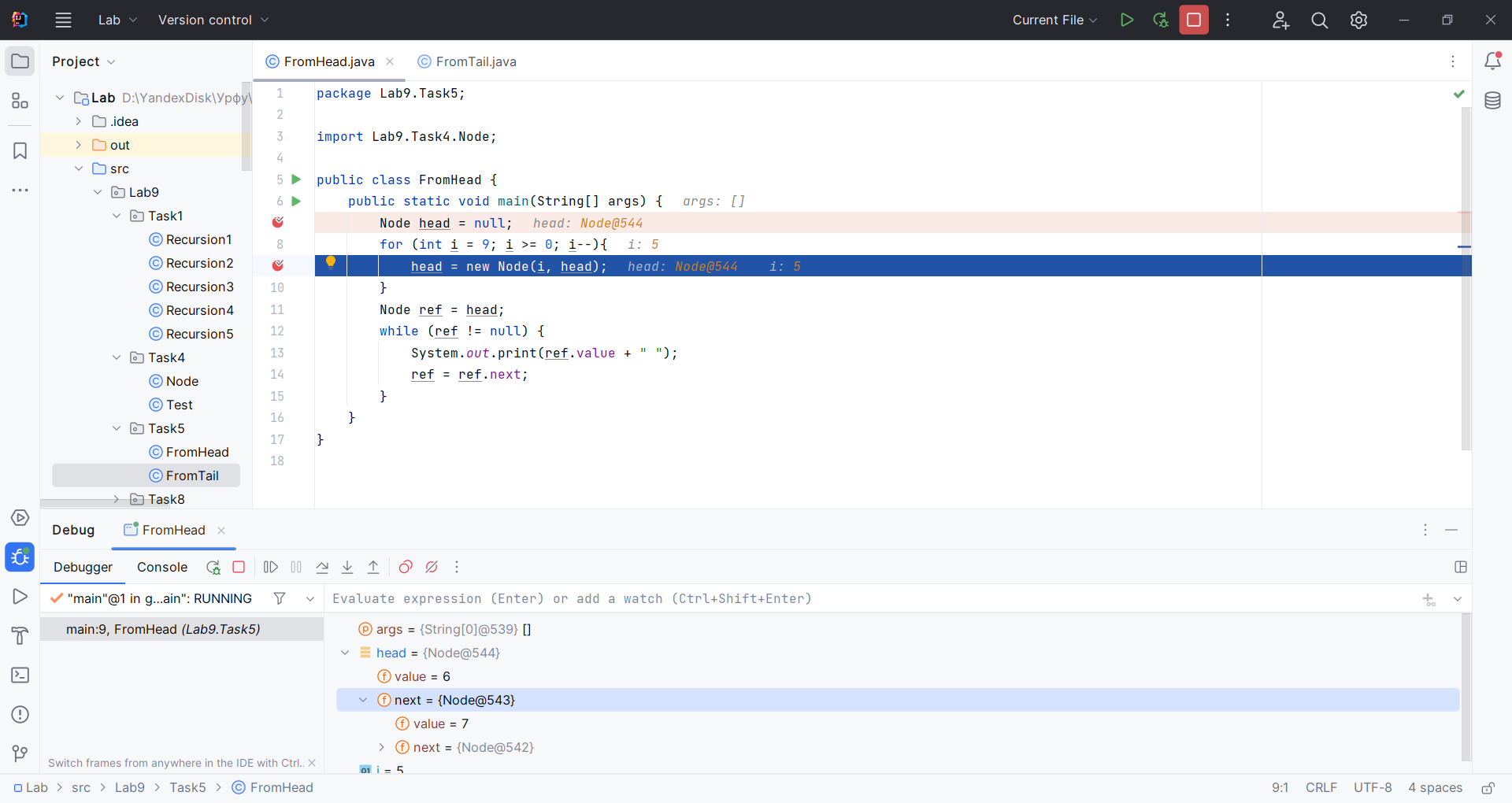


Рисунок 5 – создание с головы

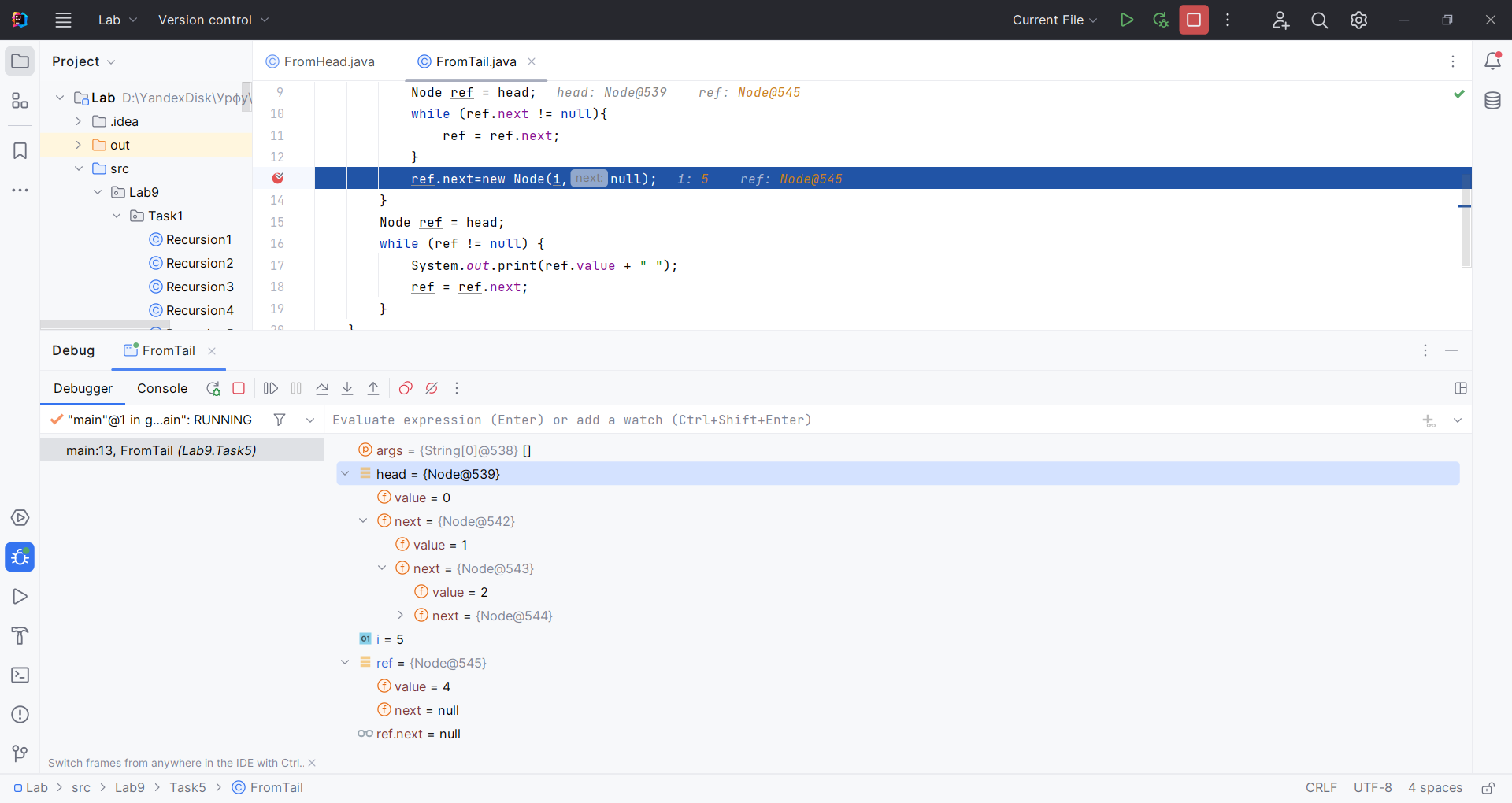


Рисунок 6 – создание с хвоста

* 1. **Заполнение HashMap 10 объектами <Integer, String>. Нахождение строк у которых ключ> 5. Если ключ = 0, вывод строки через запятую. Перемножение всех ключей, где длина строки>5.**

package Lab9;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
  
public class Task6 {  
 public static void main(String[] args) {  
 HashMap<Integer, String> objects = new HashMap<>();  
  
 for (int i = 1; i <= 10; i++) {  
 objects.put(i, "String " + i);  
 }  
  
 *//Строки у которых* ArrayList<String> keySort = new ArrayList<>();  
 for (Map.Entry<Integer, String> entry : objects.entrySet()) {  
 if (entry.getKey()>5){  
 keySort.add(entry.getValue());  
 }  
 }  
 System.*out*.println("Ключ > 5: " + keySort);  
  
 System.*out*.println();  
  
 *//Если ключ = 0, вывести строки через запятую.* System.*out*.print("Строки через запятую если ключ = 0: ");  
 if (objects.containsKey(0)) {  
 System.*out*.print(objects.values());  
 }  
 System.*out*.println();  
  
 System.*out*.println();  
  
 *//Перемножить все ключи, где длина строки>5.* Integer keyMult = null;  
 for (Map.Entry<Integer, String> entry : objects.entrySet()) {  
 if (entry.getValue().length()>5){  
 try {  
 keyMult \*= entry.getKey();  
 }catch (NullPointerException e){  
 keyMult = entry.getKey();  
 }  
 }  
 }  
 System.*out*.println("Перемножение ключей: " + keyMult);  
  
*// printHash(objects);* }  
 static void printHash(HashMap<Integer, String> map) {  
 for (Map.Entry<Integer, String> entry : map.entrySet()) {  
 System.*out*.println(String.*format*("%-15s", "Key: " + entry.getKey()) + " Value: " + entry.getValue());  
 }  
 }  
}

* 1. Две программы, моделирующие процесс, где при ведении счета по кругу вычеркивается каждый второй человек, пока не останется один. Одна из программ использует класс ArrayList, а вторая — LinkedList.

package Lab9;  
  
import java.util.ArrayList;  
import java.util.LinkedList;  
  
public class Task7 {  
 public static void main(String[] args) {  
 Integer number\_of\_people = 10;  
  
 ArrayList<String> array = new ArrayList<>();  
 LinkedList<String> linked = new LinkedList<>();  
 for (int i = 0; i < number\_of\_people; i++) {  
 array.add("Person " + i);  
 linked.add("Person " + i);  
 }  
  
 long startTime = System.*nanoTime*();  
 array = *array\_prog*(array);  
 long endTime = System.*nanoTime*();  
 long timeElapsed = endTime - startTime;  
 System.*out*.printf("ArrayList: %,.5f ms\n", timeElapsed/1\_000\_000.0);  
  
 startTime = System.*nanoTime*();  
 linked = *linked\_prog*(linked);  
 endTime = System.*nanoTime*();  
 timeElapsed = endTime - startTime;  
 System.*out*.printf("LinkedList: %,.5f ms\n", timeElapsed/1\_000\_000.0);  
 }  
  
 static ArrayList<String> array\_prog(ArrayList<String> array){  
 while (array.size() != 1){  
 for (int i = 1; i < array.size(); i+=2) {  
 array.remove(i);  
 }  
 }  
 return array;  
 }  
 static LinkedList<String> linked\_prog(LinkedList<String> array){  
 while (array.size() != 1){  
 for (int i = 1; i < array.size(); i+=2) {  
 array.remove(i);  
 }  
 }  
 return array;  
 }  
}

При выполнении данных программ LinkedList оказывается быстрее так, как при удалении изменяет только ссылку на следующий объект, в то время как ArrayList сдвигает весь список.

* 1. Проект, в котором для ввода, вывода и изменения односвязного линейного списка созданы требуемые методы.

Основной класс:

package Lab9.Task8;  
  
public class Main {  
  
 static int *offset* = 13;  
 static int *col\_offset* = 25;  
  
 static Cycle *cycle* = new Cycle();  
 static Recursion *recursion* = new Recursion();  
  
 public static void Print(String s){  
  
 System.*out*.printf("%1$-"+*offset*+"s" + "%2$-"+*col\_offset*+"s" +  
 "%1$-"+*offset*+"s" + *recursion* + "\n", s, *cycle*);  
 }  
  
 public static void main(String[] args) {  
 int tail = 2;  
 int head = 2;  
 int index = 2;  
  
 System.*out*.printf("%-"+(*offset*+*col\_offset*)+"s" + "%s\n", "CYCLE", "RECURSION");  
  
 *cycle*.createTail(tail);  
 *recursion*.createTail(tail);  
 *Print*("createTail:");  
  
 *cycle*.createHead(head);  
 *recursion*.createHead(head);  
 *Print*("createHead:");  
  
 *cycle*.createTail(tail);  
 *recursion*.createTail(tail);  
 *Print*("createTail:");  
  
 *cycle*.AddFirst();  
 *recursion*.AddFirst();  
 *Print*("AddFirst:");  
  
 *cycle*.AddLast();  
 *recursion*.AddLast();  
 *Print*("AddLast:");  
  
 *cycle*.Insert(index);  
 *recursion*.Insert(index);  
 *Print*("Insert:");  
  
 *cycle*.RemoveFirst();  
 *recursion*.RemoveFirst();  
 *Print*("RemoveFirst:");  
  
 *cycle*.RemoveLast();  
 *recursion*.RemoveLast();  
 *Print*("RemoveLast:");  
  
 *cycle*.Remove(index);  
 *recursion*.Remove(index);  
 *Print*("Remove:");  
 }  
}

1. Методы, использующие цикл

package Lab9.Task8;  
  
import Lab9.Task4.Node;  
  
public class Cycle {  
 private Node node = null;  
 private int count = 0;  
  
 private void caclCount(){  
 Node ref = node;  
 count = 0;  
 while (ref != null) {  
 count++;  
 ref = ref.next;  
 }  
 }  
 public void createHead(int lenght){  
 caclCount();  
 for (int i = lenght-1+count; i >= count; i--){  
 node = new Node(i, node);  
 }  
 }  
  
 public void createTail(int lenght){  
 caclCount();  
 for (int i = count; i < lenght + count; i++){  
 try {  
 Node ref = node;  
 while (ref.next != null){  
 ref = ref.next;  
 }  
 ref.next = new Node(i,null);  
 }catch (NullPointerException e){  
 node = new Node(i,null);  
 }  
 }  
 }  
  
 public String toString(){  
 Node ref = node;  
 String str = "";  
 while (ref != null) {  
 str += ref.value + " ";  
 ref = ref.next;  
 }  
 return str;  
 }  
  
 public void AddFirst(){  
 caclCount();  
 node = new Node(count, node);  
 }  
  
 public void AddLast(){  
 caclCount();  
 Node ref = node;  
 while (ref.next != null){  
 ref = ref.next;  
 }  
 ref.next = new Node(count,null);  
 }  
  
 public void Insert(int index){  
 caclCount();  
 Node ref = node;  
 for (int i = 1; i < index - 1; i++) {  
 ref = ref.next;  
 }  
 ref.next = new Node(count, ref.next);  
 }  
  
 public void RemoveFirst(){  
 node = node.next;  
 }  
  
 public void RemoveLast(){  
 Node ref = node;  
 while (ref.next.next != null){  
 ref = ref.next;  
 }  
 ref.next = null;  
 }  
  
 public void Remove(int index){  
 Node ref = node;  
 for (int i = 1; i < index - 2; i++) {  
 ref = ref.next;  
 }  
 ref.next = ref.next.next;  
 }  
}

1. Методы, использующие рекурсию

package Lab9.Task8;  
  
import Lab9.Task4.Node;  
  
public class Recursion {  
 private Node node = null;  
 private int count = 0;  
  
 private void caclCount(){  
 Node ref = node;  
 count = 0;  
 while (ref != null) {  
 count++;  
 ref = ref.next;  
 }  
 }  
  
 public String toString(){  
 Node ref = node;  
 return toStringRec(ref);  
 }  
 private String toStringRec(Node ref){  
 return ref == null ? "" : ref.value + " " + toStringRec(ref.next);  
 }  
  
 public void createHead(int lenght){  
 caclCount();  
 if (lenght>0){  
 node = new Node(count, node);  
 createHead(lenght - 1);  
 }  
 lenght++;  
 indexNode(lenght, node).value = lenght;  
 }  
  
 private Node lastNode(Node node){  
 if (node.next != null){  
 node = lastNode(node.next);  
 }  
 return node;  
 }  
  
 private Node indexNode(int index, Node node){  
 index--;  
 if (index > 1){  
 node = indexNode(index, node.next);  
 }  
 return node;  
 }  
  
 public void createTail(int lenght){  
 caclCount();  
 if (count < lenght + count){  
 lenght--;  
 createTail(lenght);  
 try {  
 lastNode(node).next = new Node(lenght + count, null);  
 }catch (NullPointerException e){  
 node = new Node(lenght,null);  
 }  
 }  
 }  
 public void AddFirst(){  
 caclCount();  
 node = new Node(count, node);  
 }  
  
 public void AddLast(){  
 caclCount();  
 lastNode(node).next = new Node(count,null);  
 }  
  
 public void Insert(int index){  
 caclCount();  
 Node indexNode = indexNode(index, node);  
 indexNode.next = new Node(count, indexNode.next);  
 }  
  
 public void RemoveFirst(){  
 node = node.next;  
 }  
  
 public void RemoveLast(){  
 caclCount();  
 indexNode(count, node).next = null;  
 }  
  
 public void Remove(int index){  
 caclCount();  
 Node indexNode = indexNode(index, node);  
 indexNode.next = indexNode.next.next;  
 }  
}