

Task 1. Изучить механизм интеропа между языками, попробовать у себя вызывать C/C++ (Не C++/CLI) код (суммы чисел достаточно) из Java и C#. В отчёте описать логику работы, сложности и ограничения этих механизмов.

Интероп – технология, позволяющая код, написанный на одном языке, вызывать из другого языка.

C/C++ - Java

Пишем код на C/C++ для h-файла и его реализацию. В результате мы получаем динамически подключаемую библиотеку .dll

```
#include "D:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3.2\jbr\include\jni.h"

#ifndef _Included_my_sum_Main
#define _Included_my_sum_Main
#ifdef __cplusplus
extern "C" {
#endif
JNIEXPORT jint JNICALL Java_my_sum_Main_Add
    (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif
```

*library.h*

```
#include "library.h"

JNIEXPORT jint JNICALL Java_my_sum_Main_Add
    (JNIEnv * env, jobject obj, jint a, jint b){
    int sum = (int)(a+b);
    return (jint)sum;
}
```

*library.cpp*

Получаем библиотеку libtask1\_java.dll

Пишем код на Java:

```
package my.sum;

public class Main {

    static
    {
        System.load( filename: "C:\\Users\\MairianeT\\CLionProjects\\task1-java\\cmake-build-debug\\libtask1_java.dll");
    }

    native public static int Add(int a, int b);

    public static void main(String[] args)
    {
        System.out.println(Add( a: 2, b: 3));
    }
}
```

Библиотеку загружаем по пути к файлу

C/C++ - C#

```
#ifndef TASK1_C__LIBRARY_H
#define TASK1_C__LIBRARY_H

extern "C" {
int __declspec(dllexport) Add(int, int);
}

#endif //TASK1_C__LIBRARY_H
```

*library.h*

```
#include "library.h"

int Add(int a, int b) {
    return a + b;
}
```

*library.cpp*

Реализация на C#:

```
using System.Runtime.InteropServices;

namespace Task1
{
    [1 usage]
    public class Sum
    {
        [DllImport( dllName: @"C:\Users\MairianeT\CLionProjects\task1-C#\cmake-build-debug\libtask1_C_.dll",
            CallingConvention = CallingConvention.Cdecl)]
        [1 usage]
        public static extern int Add(int a, int b);
    }
}
```

```
using System;

namespace Task1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write(Sum.Add( a: 3, b: 4));
        }
    }
}
```

Task 2. Написать немного кода на Scala и F# с использованием уникальных возможностей языка - Pipe operator, Discriminated Union, Computation expressions и т.д. Вызвать написанный код из обычных соответствующих ООП языков (Java и C#) и посмотреть, во что превращается написанный ранее код после декомпиляции в них.

Pipe operator, Discriminated Union на Scala:

```
import scala.util.chaining._
import scala.language.implicitConversions

sealed trait Shape
case class Rectangle(height: Double, width: Double) extends Shape
case class Circle(radius: Double) extends Shape
case class Square(side: Double) extends Shape
case class EquilateralTriangle(side: Double) extends Shape

object Task {
  def plus1(i: Int): Int = i + 1
  def square(i: Int): Int = i * i

  def area(shape: Shape): Double = shape match {
    case Rectangle(height, width) => height * width
    case Circle(radius) => Math.PI * radius * radius
    case Square(side) => side * side
    case EquilateralTriangle(side) => Math.sqrt(3.0/4.0) * side * side
  }

  def main(args: Array[String]): Unit = {
    val x = 1.pipe(plus1).pipe(square)
    println(x)

    val myRectangle = Rectangle(3.0, 4.0)
    println(area(myRectangle))
    val myCircle = Circle(10.0)
    println(area(myCircle))
    val mySquare = Square(5.0)
    println(area(mySquare))
    val myTriangle = EquilateralTriangle(4.0)
    println(area(myTriangle))
  }
}
```

Часть кода после декомпиляции Scala в Java:

```
public int plus1(final int i) { return i + 1; }

public int square(final int i) { return i * i; }

public double area(final Shape shape) {
    double var10000;
    if (shape instanceof Rectangle) {
        Rectangle var3 = .MODULE$.unapply((Rectangle)shape);
        double var4 = var3._1();
        double var6 = var3._2();
        var10000 = var4 * var6;
    } else if (shape instanceof Circle) {
        Circle var12 = Circle..MODULE$.unapply((Circle)shape);
        double var13 = var12._1();
        var10000 = 3.141592653589793D * var13 * var13;
    } else if (shape instanceof Square) {
        Square var17 = Square..MODULE$.unapply((Square)shape);
        double var18 = var17._1();
        var10000 = var18 * var18;
    } else {
        if (!(shape instanceof EquilateralTriangle)) {
            throw new MatchError(shape);
        }

        EquilateralTriangle var22 = EquilateralTriangle..MODULE$.unapply((EquilateralTriangle)shape);
        double var23 = var22._1();
        var10000 = Math.sqrt(0.75D) * var23 * var23;
    }

    return var10000;
}
```

## Pipe operator, Discriminated Union, Competition expression на F#

```
// pipe operator
int -> int
let plus1 x = x + 1
int -> int
let square x = x * x
int
let a =
    1 : int
    |> plus1 : int
    |> square
printfn $"%i{a}"
```

```
// Discriminated Union
type Shape =
    | Rectangle of double * double
    | Circle of double
    | Square of double
    | EquilateralTriangle of double
float
let pi = 3.14159
Shape -> double
let area myShape =
    match myShape with
    | Rectangle (h, w) -> h * w
    | Circle radius -> pi * radius * radius
    | Square s -> s * s
    | EquilateralTriangle s -> sqrt(3.0/4.0) * s * s

let height, width = 4.0, 10.0
Shape
let myRectangle = Rectangle(height, width)
printfn $"Rectangle area is %f{area myRectangle}"

float
let radius = 10.0
Shape
let myCircle = Circle(radius)
printfn $"Circle area is %f{area myCircle}"

float
let squareSide = 5.0
Shape
let mySquare = Square(squareSide)
printfn $"Square area is %f{area mySquare}"
```

```
// Competition expression
(int * int * int * int * int * int * int * int * int * int) list
let list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
seq<int>
let doubles =
    seq {
        for i in 1..10
            -> i * 2
    }
for db in doubles do
    printfn $"{i}{db}"
```

Часть кода после декомпиляции в C#

```
[Serializable]
[StructLayout(LayoutKind.Auto, CharSet = CharSet.Auto)]
[DebuggerDisplay("{__DebugDisplay(),nq}")]
[CompilationMapping(SourceConstructFlags.SumType)]
public abstract class Shape : IEquatable<Shape>, IStructuralEquatable, IComparable
{
    public static class Tags
    {
        public const int Rectangle = 0;

        public const int Circle = 1;

        public const int Square = 2;

        public const int EquilateralTriangle = 3;
    }

    [Serializable]
    [SpecialName]
    [DebuggerTypeProxy(typeof(Rectangle@DebugTypeProxy))]
    [DebuggerDisplay("{__DebugDisplay(),nq}")]
    public class Rectangle : Shape
    {
        [DebuggerBrowsable(DebuggerBrowsableState.Never)]
        [CompilerGenerated]
        [DebuggerNonUserCode]
        internal readonly double item1;

        [DebuggerBrowsable(DebuggerBrowsableState.Never)]
        [CompilerGenerated]
        [DebuggerNonUserCode]
        internal readonly double item2;

        [CompilationMapping(SourceConstructFlags.Field, 0, 0)]
        [CompilerGenerated]
        [DebuggerNonUserCode]
        internal readonly double item3;
```

## Pipe operator

```
public static int plus1(int x)
{
    return x + 1;
}

public static int square(int x)
{
    return x * x;
}
```

## Discriminated Union

```
public static double area(Shape myShape)
{
    switch (myShape.Tag)
    {
        default:
        {
            Shape.Rectangle rectangle = (Shape.Rectangle)myShape;
            return rectangle.item1 * rectangle.item2;
        }
        case 1:
        {
            Shape.Circle circle = (Shape.Circle)myShape;
            double item = circle.item;
            return 3.14159 * item * item;
        }
        case 2:
        {
            Shape.Square square = (Shape.Square)myShape;
            double item = square.item;
            return item * item;
        }
        case 3:
        {
            Shape.EquilateralTriangle equilateralTriangle = (Shape.Equilatera
            double item = equilateralTriangle.item;
            return Math.Sqrt(3.0 / 4.0) * item * item;
        }
    }
}
```

## Competition Expression

```
[Serializable]
[SpecialName]
[StructLayout(LayoutKind.Auto, CharSet = CharSet.Auto)]
[CompilationMapping(SourceConstructFlags.Closure)]
internal sealed class doubles@44 : GeneratedSequenceBase<int>
{
    [DebuggerBrowsable(DebuggerBrowsableState.Never)]
    [CompilerGenerated]
    [DebuggerNonUserCode]
    public IEnumerator<int> @enum;

    [DebuggerBrowsable(DebuggerBrowsableState.Never)]
    [CompilerGenerated]
    [DebuggerNonUserCode]
    public int pc;

    [DebuggerBrowsable(DebuggerBrowsableState.Never)]
    [CompilerGenerated]
    [DebuggerNonUserCode]
    public int current;

    public doubles@44(IEnumerator<int> @enum, int pc, int current)
    {
        this.@enum = @enum;
        this.pc = pc;
        this.current = current;
        base..ctor();
    }

    public override int GenerateNext(ref IEnumerable<int> next)
    {
        switch (pc)
        {
            default:
                @enum = Operators.OperatorIntrinsics.RangeInt32(1, 1, 10).Get
                pc = 1;
                goto case 2;
            case 2:
                // ...
        }
    }
}
```

C# и Java не имеют аналогов уникальных возможностей языков F# и Scala, в результате чего код на них получается довольно объемным



Task 3. Написать алгоритм обхода графа (DFS и BFS) на языке Java, собрать в пакет и опубликовать (хоть в Maven, хоть в Gradle, не имеет значения). Использовать в другом проекте на Java/Scala этот пакет. Повторить это с C#/F#. В отчёте написать про алгоритм работы пакетных менеджеров, особенности их работы в C# и Java мирах.

Алгоритмы DFS и BFS на Java:

```
package com.company;

import java.io.*;
import java.util.*;

public class Graph {
    public int V;
    public LinkedList<Integer> adj[];

    public Graph(int v)
    {
        V = v;
        adj = new LinkedList[V];
        for (int i=0; i<V; ++i)
            adj[i] = new LinkedList();
    }

    public void addEdge(int v, int w) { adj[v].add(w); }
```

```
public void BFS(int s)
{
    boolean visited[] = new boolean[V];
    LinkedList<Integer> queue = new LinkedList<Integer>();
    visited[s]=true;
    queue.add(s);

    while (queue.size() != 0)
    {
        s = queue.poll();
        System.out.print(s+" ");

        Iterator<Integer> i = adj[s].listIterator();
        while (i.hasNext())
        {
            int n = i.next();
            if (!visited[n])
            {
                visited[n] = true;
                queue.add(n);
            }
        }
    }
}
```

```

public void DFSUtil(int v, boolean visited[])
{
    visited[v] = true;
    System.out.print(v + " ");

    Iterator<Integer> i = adj[v].listIterator();
    while (i.hasNext()) {
        int n = i.next();
        if (!visited[n])
            DFSUtil(n, visited);
    }
}

public void DFS(int v)
{
    boolean visited[] = new boolean[V];
    DFSUtil(v, visited);
}

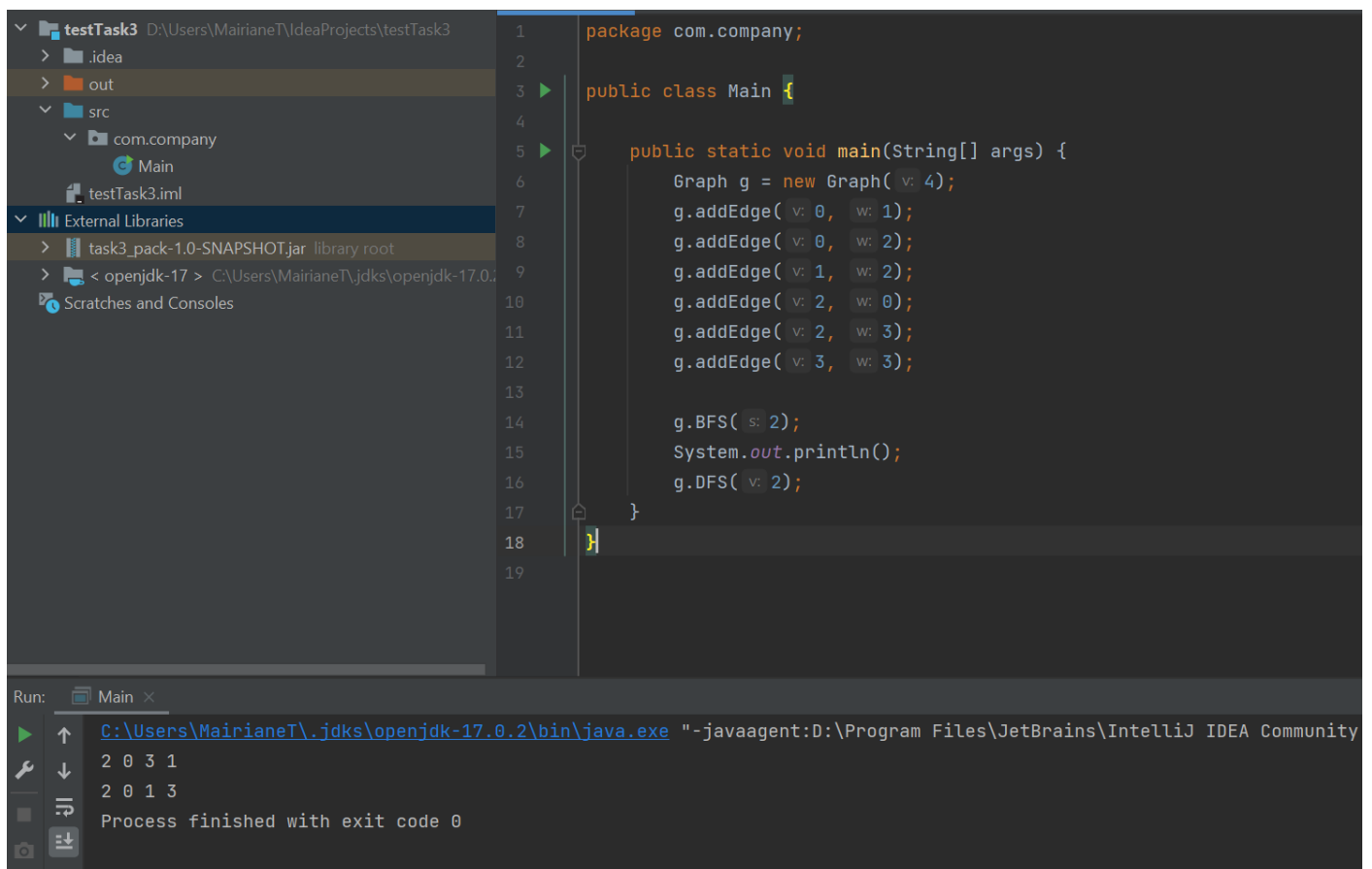
```

Создадим пакет в Maven



Получили jar файл, который можно использовать в других проектах на Scala и Java

## Java



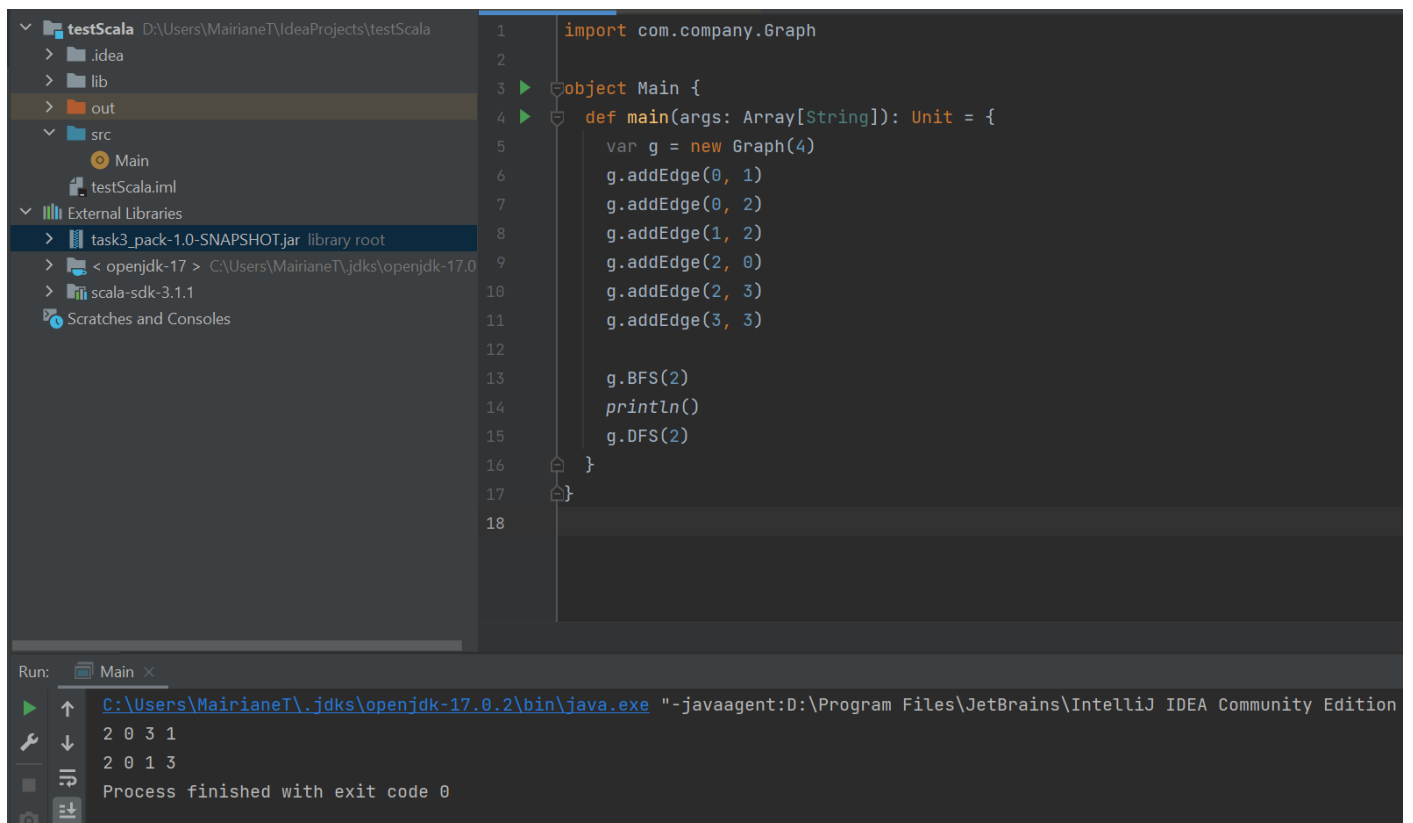
The screenshot shows the IntelliJ IDEA IDE with a Java project named 'testTask3'. The project structure on the left includes 'src' with a 'com.company' package containing a 'Main' class. The 'External Libraries' section shows 'task3\_pack-1.0-SNAPSHOT.jar' as the library root. The main editor displays the following Java code:

```
1 package com.company;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Graph g = new Graph(4);
7         g.addEdge(0, 1);
8         g.addEdge(0, 2);
9         g.addEdge(1, 2);
10        g.addEdge(2, 0);
11        g.addEdge(2, 3);
12        g.addEdge(3, 3);
13
14        g.BFS(2);
15        System.out.println();
16        g.DFS(2);
17    }
18 }
```

The Run window at the bottom shows the execution of 'Main' using 'C:\Users\MairianeT\jdk\openjdk-17.0.2\bin\java.exe'. The output is:

```
2 0 3 1
2 0 1 3
Process finished with exit code 0
```

## Scala



The screenshot shows the IntelliJ IDEA IDE with a Scala project named 'testScala'. The project structure on the left includes 'src' with a 'Main' object. The 'External Libraries' section shows 'task3\_pack-1.0-SNAPSHOT.jar' as the library root. The main editor displays the following Scala code:

```
1 import com.company.Graph
2
3 object Main {
4     def main(args: Array[String]): Unit = {
5         var g = new Graph(4)
6         g.addEdge(0, 1)
7         g.addEdge(0, 2)
8         g.addEdge(1, 2)
9         g.addEdge(2, 0)
10        g.addEdge(2, 3)
11        g.addEdge(3, 3)
12
13        g.BFS(2)
14        println()
15        g.DFS(2)
16    }
17 }
```

The Run window at the bottom shows the execution of 'Main' using 'C:\Users\MairianeT\jdk\openjdk-17.0.2\bin\java.exe'. The output is:

```
2 0 3 1
2 0 1 3
Process finished with exit code 0
```

## Алгоритмы DFS и BFS на C#

```
namespace task3
{
    public class Graph
    {
        private int _v;
        private List<int>[] adj;

        public Graph(int v)
        {
            _v = v;
            adj = new List<int>[v];
            for (int i = 0; i < v; i++)
            {
                adj[i] = new List<int>();
            }
        }

        public void AddEdge(int v, int w)
        {
            adj[v].Add(w);
        }
    }
}
```

2 usages

```
public void DFSUtil(int v, bool[] visited)
{
    visited[v] = true;
    Console.Write(v + " ");
    List<int> vList = adj[v];
    foreach (var n in vList)
    {
        if (!visited[n])
            DFSUtil(n, visited);
    }
}

public void DFS(int v)
{
    bool[] visited = new bool[_v];
    DFSUtil(v, visited);
}
```

```

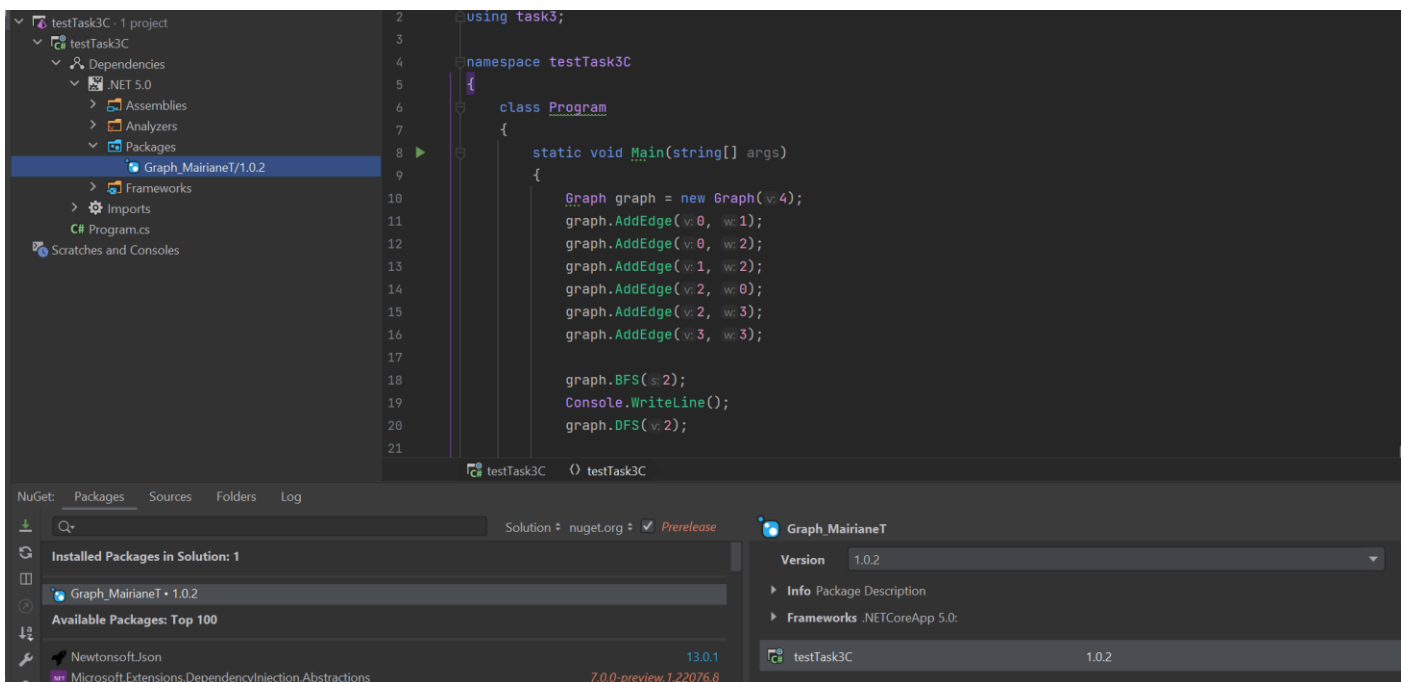
public void BFS(int s)
{
    bool[] visited = new bool[_v];
    for(int i = 0; i < _v; i++)
        visited[i] = false;
    LinkedList<int> queue = new LinkedList<int>();
    visited[s] = true;
    queue.AddLast(s);

    while(queue.Any())
    {
        s = queue.First();
        Console.Write(s + " ");
        queue.RemoveFirst();
        var list = adj[s];

        foreach (var val:int in list.Where(val:int => !visited[val]))
        {
            visited[val] = true;
            queue.AddLast(val);
        }
    }
}

```

Публикуем проект на сайте nuget, далее можем найти пакет в Rider и подключить



The screenshot shows the JetBrains Rider IDE interface. On the left, the 'Solution Explorer' displays the project structure for 'testTask3F'. It includes dependencies for '.NET 5.0', 'FSharp.Core/5.0.0', and 'Graph\_MairianeT/1.0.2'. The main editor area shows the 'open tasks' file with the following F# code:

```
2 open tasks
3
4 Graph
5
6 let graph = new Graph(4)
7 graph.AddEdge(0, 1);
8 graph.AddEdge(0, 2);
9 graph.AddEdge(1, 2);
10 graph.AddEdge(2, 0);
11 graph.AddEdge(2, 3);
12 graph.AddEdge(3, 3);
13
14 graph.BFS(2)
15 printfn ""
16 graph.DFS(2)
```

At the bottom, the 'Run' console shows the execution command and output:

```
Run: "D:\Program Files\JetBrains\JetBrains Rider 2021.3.3\plugins\dpa\DotFiles\JetBrains.DPA.Runner.exe" --hand
:/Users/MairianeT/RiderProjects/testTask3F/testTask3F/bin/Debug/net5.0/testTask3F.exe
2 0 3 1
2 0 1 3
Process finished with exit code 0.
```

Task 4. Изучить инструменты для оценки производительности в C# и Java. Написать несколько алгоритмов сортировок (и взять стандартную) и запустить бенчмарки (в бенчмарках помимо времени выполнения проверить аллокации памяти). В отчёт написать про инструменты для бенчмаркинга, их особенности, анализ результатов проверок.

C#

The screenshot shows a C# code snippet for a sorting benchmark. It defines a namespace 'task4' and a class 'Sorts' with a 'Setup' method and two benchmark methods: 'MergeSort' and 'InsertionSort'.

```
namespace task4
{
    [MemoryDiagnoser]
    [Usage]
    public class Sorts
    {
        [Params(100, 10000)]
        public int N;
        public int[] array;

        [GlobalSetup]
        public void Setup()
        {
            array = new int[N];
            var random = new Random();
            for (var i = 0; i < array.Length; i++)
            {
                array[i] = random.Next();
            }
        }

        [Benchmark]
        public int[] MergeSort()
        {
            return MergeSort(array, lowIndex: 0, highIndex: array.Length - 1);
        }

        [Benchmark]
        public void InsertionSort()
        {
            int x;
```

Когда запускаем класс, BenchmarkDotNet сначала выполняет подготовку: определяет количество итераций и оценивает накладную работу, а затем переходит к разогреву и измерению.

Method	N	Mean	Error	StdDev	Gen 0	Allocated
MergeSort	100	2,425.6 ns	59.05 ns	156.60 ns	1.2474	5,224 B
InsertionSort	100	114.4 ns	2.24 ns	4.93 ns	-	-
StandartSort	100	201.7 ns	4.05 ns	4.16 ns	-	-
MergeSort	10000	351,771.1 ns	5,821.71 ns	5,445.63 ns	188.9648	792,040 B
InsertionSort	10000	10,349.0 ns	120.79 ns	112.99 ns	-	-
StandartSort	10000	48,687.7 ns	942.38 ns	925.55 ns	-	-

## Java

```
package com.company;

import org.openjdk.jmh.annotations.*;

import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.TimeUnit;

@State(Scope.Benchmark)
public class BenchmarkSorts {
    @Param({"100", "10000"})
    private int N;
    private int[] array;
    @Setup(Level.Invocation)
    public void setUp() {
        array = new int[N];
        Random rand = new Random();
        for (int i = 0; i < array.length; i++) {
            array[i] = rand.nextInt();
        }
    }

    @Benchmark
    @BenchmarkMode(Mode.AverageTime)
    @OutputTimeUnit(TimeUnit.NANOSECONDS)
    @Fork(value = 1)
    @Measurement(iterations = 10)
    @Warmup(iterations = 1)
    public void BubbleSort() { Sorts.bubbleSort(array); }

    @Benchmark
    @BenchmarkMode(Mode.AverageTime)
    @OutputTimeUnit(TimeUnit.NANOSECONDS)
    @Fork(value = 1)
    @Measurement(iterations = 10)
    @Warmup(iterations = 1)
```

Аналогично проводим измерения (используем Java Microbenchmark Harness (JMH))

Benchmark	(N)	Mode	Cnt	Score	Error	Units
BenchmarkSorts.BubbleSort	100	avgt	10	24587,005 ±	17407,109	ns/op
BenchmarkSorts.BubbleSort	10000	avgt	10	171289006,364 ±	138553516,453	ns/op
BenchmarkSorts.InsertionSort	100	avgt	10	2797,572 ±	2430,571	ns/op
BenchmarkSorts.InsertionSort	10000	avgt	10	10903820,607 ±	8731229,849	ns/op
BenchmarkSorts.StandardSort	100	avgt	10	3102,578 ±	2324,688	ns/op
BenchmarkSorts.StandardSort	10000	avgt	10	630611,100 ±	498887,891	ns/op

С помощью бенчмарков можно получить представление о работе сортировок: их эффективности, используемой ими памяти

Task 5. Используя инструменты dotTrace, проанализировать работу написанного кода для бэкапов. Необходимо написать сценарий, когда в цикле будет выполняться много запусков, будут создаваться и удаляться точки. Проверить два сценария: с реальной работой с файловой системой и без неё. В отчёте необходимо проанализировать полученные результаты, сделать вывод о написанном коде.

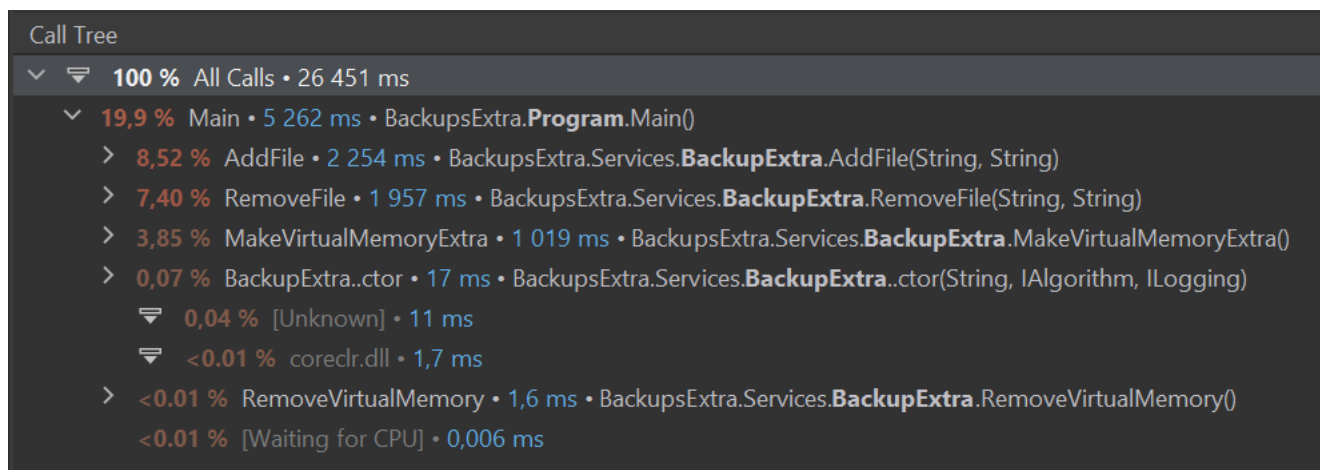
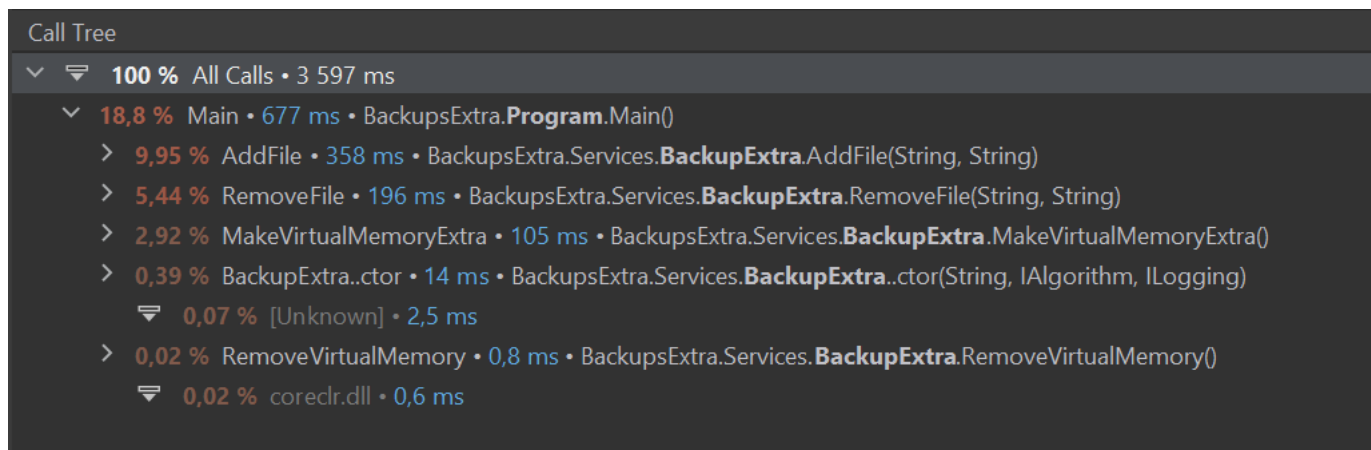
Создаем цикл, который 100/1000 раз создает точки, добавляет в них файлы, удаляет точки.

100 % All Calls • 6 262 ms
20,7 % Main • 1 298 ms • BackupsExtra. <b>Program</b> .Main()
8,30 % MakeRestorePointExtra • 520 ms • BackupsExtra.Services. <b>BackupExtra</b> .MakeRestorePointExtra()
5,44 % AddFile • 341 ms • BackupsExtra.Services. <b>BackupExtra</b> .AddFile(String, String)
2,94 % RemoveFile • 184 ms • BackupsExtra.Services. <b>BackupExtra</b> .RemoveFile(String, String)
2,24 % SetNumberLimit • 140 ms • BackupsExtra.Services. <b>BackupExtra</b> .SetNumberLimit(Int32)
1,48 % PushToNumberLimit • 93 ms • BackupsExtra.Services. <b>BackupExtra</b> .PushToNumberLimit()
0,30 % BackupExtra..ctor • 19 ms • BackupsExtra.Services. <b>BackupExtra</b> ..ctor(String, IAlgorithm, ILogging)
0,03 % coreclr.dll • 1,6 ms
<0.01 % [Unknown] • 0,5 ms

Call Tree
100 % All Calls • 51 856 ms
20,3 % Main • 10 522 ms • BackupsExtra. <b>Program</b> .Main()
6,49 % MakeRestorePointExtra • 3 364 ms • BackupsExtra.Services. <b>BackupExtra</b> .MakeRestorePointExtra()
4,95 % AddFile • 2 569 ms • BackupsExtra.Services. <b>BackupExtra</b> .AddFile(String, String)
3,72 % RemoveFile • 1 927 ms • BackupsExtra.Services. <b>BackupExtra</b> .RemoveFile(String, String)
2,99 % SetNumberLimit • 1 549 ms • BackupsExtra.Services. <b>BackupExtra</b> .SetNumberLimit(Int32)
1,90 % PushToNumberLimit • 987 ms • BackupsExtra.Services. <b>BackupExtra</b> .PushToNumberLimit()
0,18 % BackupExtra..ctor • 93 ms • BackupsExtra.Services. <b>BackupExtra</b> ..ctor(String, IAlgorithm, ILogging)
0,06 % [Unknown] • 31 ms
<0.01 % coreclr.dll • 1,4 ms



## Без файловой системы



При использовании файловой системы основные затраты идут на архивацию, добавление и удаление файлов.

Без файловой системы — на добавление и удаление файлов.