Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

 «Магнитогорский государственный технический университет им. Г. И. Носова**»**

(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Кафедра вычислительной техники и программирования

Лабораторная работа №5

по дисциплине «Метрология и стандартизация программного обеспечения»

## название лабораторной работы: «Оценка качества программы с помощью объектно-ориентированных метрик. Часть 2»

Исполнитель: Комлев Н.Б., студент 3 курса, группа АВб-19-1

Руководитель: Сибилева Н.С., ст. преподаватель каф. ВТиП

Магнитогорск, 2022

**СОДЕРЖАНИЕ**

[1 Постановка задания 3](#_Toc97064744)

[2 Выполнения задания 3](#_Toc97064745)

[Заключение 7](#_Toc97064746)

# Постановка задания

Базовый класс – файл. Производные – звуковой файл и видео-файл. Создать класс Каталог, который может содержать оба вида объектов. Предусмотреть метод подсчета отдельно звуковых и видео-файлов (использовать оператор instanceof).

# Выполнения задания

**Программа** была написана на языке программирования JavaScript (Node.js) и предоставлена ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148 | **class** File {  constructor(name = 'New file.txt', length = 0) {  **this**.setFullFileName(name)  **this**.creationTime = **new** Date();  **this**.lastUpdateTime = **new** Date();  **this**.length = length;  }  setName(newName) {  **this**.name = newName;  **this**.setLastUpdateTime(**new** Date());  }  setExtension(newExtension) {  **this**.extension = newExtension;  **this**.setLastUpdateTime(**new** Date());  }  setFullFileName(newName) {  **const** splitted = newName.split(**'.'**);  **this**.extension = `.${splitted[splitted.length - 1]}`;  **this**.name = newName.substr(0, newName.length - **this**.extension.length,);  **this**.setLastUpdateTime(**new** Date());  }  setLastUpdateTime(newTime) {  **this**.lastUpdateTime = newTime;  }  setLength(newLength) {  **this**.length = newLength;  }  getFileName() {  **return** `${**this**.name}${**this**.extension}`;  }  getLength() {  **return** **this**.length;  }  getCreationTime() {  **return** **this**.creationTime;  }  getLastUpdateTime() {  **return** **this**.lastUpdateTime;  }  }  **class** AudioFile extends File {  constructor(name = 'New audio file.mp3', fileLength = 0, audioLength = 0) {  **const** splitted = name.split(**'.'**);  **if** (splitted[splitted.length-1] === **"wav"** || splitted[splitted.length-1] === **"mp3"** || splitted[splitted.length-1] === **"wma"** ||  splitted[splitted.length-1] === **"ogg"** || splitted[splitted.length-1] === **"kar"**) {  super(name, fileLength);  **this**.audioLength = audioLength;  } **else** {  **throw** **new** Error(`${name} is not an audio file`);  }  }  setAudioLength(length) {  **this**.audioLength = length;  **this**.setLastUpdateTime(**new** Date());  }  getAudioLength() {  **return** **this**.audioLength;  }  }  **class** VideoFile extends File {  constructor(name = 'New video file.avi', fileLength = 0, videoLength = 0) {  **const** splitted = name.split(**'.'**);  **if** (splitted[splitted.length-1] === **"avi"** || splitted[splitted.length-1] === **"mp4"**  || splitted[splitted.length-1] === **"mov"** || splitted[splitted.length-1] === **"mvk"**) {  super(name, fileLength);  **this**.videoLength = videoLength;  } **else** {  **throw** **new** Error(`${name} is not an video file`);  }  }  setVideoLength(length) {  **this**.videoLength = length;  **this**.setLastUpdateTime(**new** Date());  }  getVideoLength() {  **return** **this**.videoLength;  }  }  **class** Catalog {  constructor(files = []) {  **this**.files = files;  }  addFile(file) {  **this**.files.push(file);  }  getAudioFilesCount() {  let audio = 0;  **for** (let file of **this**.files) {  **if** (file instanceof AudioFile) {  audio++;  }  }  **return** audio;  }  getVideoFilesCount() {  let video = 0;  **for** (let file of **this**.files) {  **if** (file instanceof VideoFile) {  video++;  }  }  **return** video;  }  getCatalogLength() {  **return** **this**.files.length;  }  getCatalog() {  **return** **this**.files;  }  }  **const** main = () => {  **try** {  **const** catalog = **new** Catalog();  catalog.addFile(**new** AudioFile('Me.mp3'));  catalog.addFile(**new** VideoFile('Me.mp4'));  catalog.addFile(**new** VideoFile('Me1.mp4'));  catalog.addFile(**new** VideoFile('Me2.mp4'));  console.log(catalog.getCatalog());  console.log(`Кол-во аудиофайлов: ${catalog.getAudioFilesCount()}`);  console.log(`Кол-во видеофайлов: ${catalog.getVideoFilesCount()}`);  } **catch** (err) {  console.log(err.message);  }  }  main(); |

Результат выполнения данной программы представлен на рисунке 1.

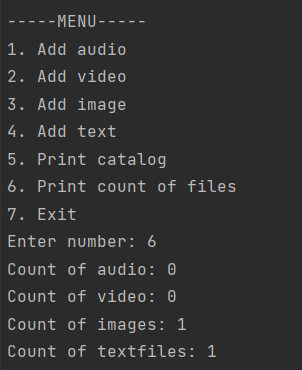


Рисунок 1 – Результат выполнения программы

Разработанный алгоритм решения задачи, представлен на рисунке 2.

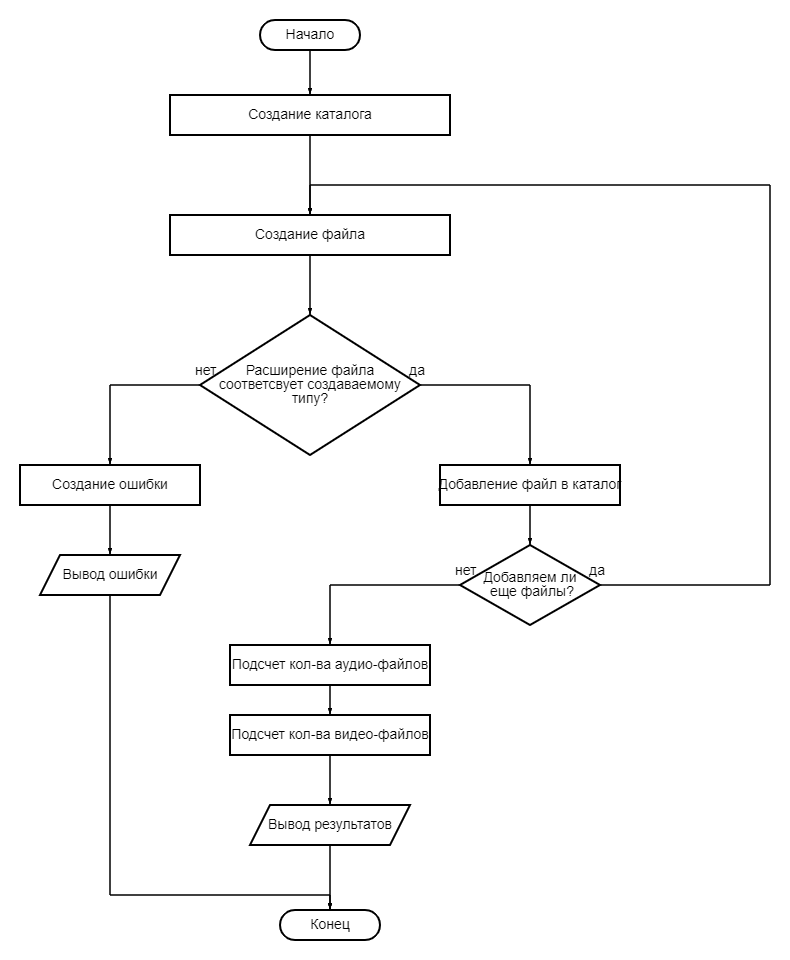


Рисунок 2 – Алгоритм решения задачи

**Метрики Лоренца и Кидда:**

**Метрика 1: Размер класса CS (Class Size)**

Большие значения CS указывают, что класс имеет слишком много обязанностей. Они уменьшают возможность повторного использования класса, усложняют его реализацию и тестирование. Чем меньше среднее значение размера, тем больше вероятность повторного использования класса.

File : CS = 14.

Catalog: CS = 9.

AudioFile: CS = 17.

VideoFile: CS = 17.

Image: CS = 14.

TextImage: CS = 17.

**Метрика 2: Количество операций, переопределяемых подклассом, NOO (Number of Operations Overridden by a Subclass)**

Переопределением называют случай, когда подкласс замещает операцию, унаследованную от суперкласса, своей собственной версией. Большие значения NOO обычно указывают на проблемы проектирования.

NOO = 0;

**Метрика 3: Количество операций, добавленных подклассом, NOA (Number of Operations Added by a Subclass)**

AudioFile: NOA = 2

VideoFile: NOA = 2

TextFile: NOA = 2

**Метрика 4: Индекс специализации SI (Specialization Index)**

Обеспечивает грубую оценку степени специализации каждого подкласса. Специализация достигается добавлением, удалением или переопределением операций:

SI = (NOO x уровень) /Mобщ,

где уровень — номер уровня в иерархии, на котором находится подкласс, Мобщ — общее количество методов класса. Чем выше значение SI, тем больше вероятность того, что в иерархии классов есть классы, нарушающие абстракцию суперкласса.

SI = 0

**Метрика 5: Средний размер операции OSAVG (Average Operation Size)**

В качестве индикатора размера может использоваться количество строк программы, однако LOC-оценки приводят к известным проблемам.

File : OSAVG = 47

AudioFile: OSAVG = 18

VideoFile: OSAVG = 18

Image: OSAVG = 8

TextFile: OSAVG = 18

Catalog: OSAVG = 54

**Метрика 6: Сложность операции ОС (Operation Complexity)**

|  |  |
| --- | --- |
| Параметр | Вес |
| Вызовы функций API | 5 |
| Присваивания | 0,5 |
| Арифметические операции | 2 |
| Сообщения с параметрами | 3 |
| Параметры | 0,3 |
| Временные переменные | 0,5 |

Значение метрики OC вычисляется суммированием оценок с весовыми коэффициентами, приведенными в таблице.

File : OС = 15\*0,5 + 2\*2 + 3\*3 + 4 \* 0.3 + 0.5 = 22.2

AudioFile: OС = 5\*0,5 + 5\*2 + 7\*3 + 8\*0.3 + 0.5 = 36.4

VideoFile: OС = 5\*0,5 + 4\*2 + 6\*3 + 7\*0.3 + 0.5 = 31.1

Image: OС = 4\*0,5 + 3\*2 + 5\*3 + 6\*0.3 + 0.5 = 25.8

TextFile: OС = = 5\*0,5 + 3\*2 + 5\*3 + 6\*0.3 + 0.5 = 26.3

**Метрика 7: Среднее количество параметров на операцию NPAVG (Average Number of Parameters per operation)**

Чем больше параметров у операции, тем сложнее сотрудничество между объектами. Поэтому значение NPAVG должно быть как можно меньшим.

Заключение

Программа реализована на языке JavaScript. Базовый класс – файл. Производные – звуковой файл и видео-файл. Создан класс Каталог, который может содержать оба вида файлов. Предусмотрен метод подсчета аудио и видео файлов(использовать оператор instanceof).

В работе разработаны блок-схема работы программы. Проведен анализ разработанной программы с помощью метрик Лоренца и Кидда.