# Отчет по лабораторной работе №3

# 1 Цель и постановка задачи

В ходе выполнения лабораторной работы №3 необходимо выполнить:

1. Реализовать свой загрузчик классов.
2. Продемонстрировать его работу.
3. Продемонстрировать разницу между своей реализацией и стандартной.

# 2 Разработка

ClassLoader является «загрузчиком классов». Специальная функция заключается в загрузке файла класса в виртуальную машину по мере выполнения программы.

Любой программный код, связанный с Java, неотделим от ClassLoader. Например, при запуске JVM ему необходимо загрузить некоторые базовые загрузочные файлы .class, загрузить базовые файлы, такие как String или Integer.class. Однако при запуске JVM загружает не все файлы классов одновременно, а динамически подгружает их по мере необходимости.

## 2.1 Разработка загрузчика классов

Предположим, нам нужен пользовательский загрузчик классов, путь загрузки по умолчанию - пакет jar и ресурсы в папке:

'C:\Users\d4nyi\.m2\repository\lab_4\src\main\java'.

Мы определили метод поиска класса в методе findClass(), а затем данные сгенерировали в объект Class с помощью defineClass().

```
import java.io.*;

public class DiskClassLoader extends ClassLoader {

    private String mLibPath;

    public DiskClassLoader(String path) {
        mLibPath = path;
    }

    @Override
    protected Class<?> findClass(String name) throws ClassNotFoundException {

        String fileName = getFileName(name);
```

```
        File file = new File(mLibPath,fileName);

        try {
            FileInputStream is = new FileInputStream(file);

            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            int len = 0;
            try {
                while ((len = is.read()) != -1) {
                    bos.write(len);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

            byte[] data = bos.toByteArray();
            is.close();
            bos.close();

            //Преобразовывает двоичное содержимое класса в объект Class.
            //Если он не соответствует требованиям, он вызывает различные исключения.
            return defineClass(name,data,0,data.length);

        } catch (IOException e) {

            e.printStackTrace();
        }

        return super.findClass(name);
    }

    // Получить имя файла класса для загрузки
    private String getFileName(String name) {

        int index = name.lastIndexOf('.');
        if(index == -1){
            return name+".class";
        }else{
            return name.substring(index+1)+".class";
        }
    }

}
```

## 2.2 Демонстрация работы

Теперь создадим Test.java и Say.java для проверки корректности работы:

```
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Test {

    public static void main(String arg[]) {

        DiskClassLoader diskLoader =
            new
DiskClassLoader("C:\\Users\\d4nyi\\.m2\\repository\\lab_4\\src\\main\\java");

        try {
            // Загружаем файл класса
            Class c = diskLoader.loadClass("Say");

            if(c != null){
                try {
                    Object obj = c.newInstance();
                    Method method = c.getDeclaredMethod("say",null);
                    // Вызываем метод say класса Test через отражение
                    method.invoke(obj, null);
                } catch (InstantiationException | IllegalAccessException
                        | NoSuchMethodException
                        | SecurityException |
                        IllegalArgumentException |
                        InvocationTargetException e) {
                    e.printStackTrace();
                }
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

}
```

```
public class Say {

    public void say(){
        System.out.println("Say Hello");
    }

}
```

Результат выполнения следующий:

```
Say Hello

Process finished with exit code 0
```