МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

САНКТ**-**ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**,** МЕХАНИКИ И ОПТИКИ

Кафедра Информатики и прикладной математики

Лабораторная работа №4 (8 вариант)

по предмету «Верификация моделей программ»

Выполнил:

студент гр. № Р4115

Назукин Д. Е.

Проверил :

Кореньков Ю. Д.

Санкт-Петербург

2017

**Цели**

Освоение верификации простых манипуляций со структурами данных с учетом модели типизации.

**Задачи**

Создать библиотеку, позволяющую сохранять в бинарный файл и загружать из него набор закодированных линейным кодом в соответствии с предыдущим заданием функций или методов. Изменить имеющуюся программу так, чтобы она, кроме входных файлов и имени выходной директории, принимала через аргументы командной строки имя для выходного файла с бинарным представлением, формируемого с помощью созданной библиотеки. Дополнить входной язык возможностью определения пользовательских типов данных, обладающих набором полей и методов.

**Описание работы**

Пользовательские типы данных позволяют создавать пользователю собственные типы данных (классы) отличных от стандартных. Класс может обладать набором полей и методов с определенными модификаторами доступа (*public*, *private*). *Public* означает что метод или поле доступно к доступу вне класса, *private* же означает что метод или поле будет доступно только внутри реализации класса. Каждый класс должен обладать конструктором, который инициализирует новый член класса. Члены класса создаются с помощью оператора *new.*

**Аспекты реализации**

Для сохранения в бинарный файл было использовано Java Serialization API. Класс сериализуемого объекта должен реализовывать интерфейс *java.io.Serializable:*

**public class** MethodDefinition **implements** Serializable

Запись и чтение из файла происходит следующим образом:

**public void** printBinCodes(Map<MethodSignature, MethodDefinition> binFuncs, Map<String, ClassDefinition> binClasses, String outputPath) **throws** IOException {  
 File file = **new** File(outputPath);  
 file.getParentFile().mkdirs();  
 **try** (ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream(file))) {  
 List<Object> list = **new** ArrayList<>();  
 list.add(binFuncs);  
 list.add(binClasses);  
 oos.writeObject(list);  
 oos.flush();  
 } **catch** (Exception e) {  
 **throw** e;  
 }  
}

**public** List<Object> readBinCodes(String path) **throws** IOException, ClassNotFoundException {  
 **try** (ObjectInputStream oin = **new** ObjectInputStream(**new** FileInputStream(path))) {  
 **return** (List<Object>) oin.readObject();  
 } **catch** (Exception e) {  
 **throw** e;  
 }  
}

Для хранения структуры класса используется следующий класс:

**public class** ClassDefinition **implements** Serializable {  
 **private** String **className**;  
  
 **private** Map<String, Integer> **fields** = **new** LinkedHashMap<>();  
 **private** Map<String, Modificator> **fieldsModificator** = **new** LinkedHashMap<>();  
  
 **private** Map<MethodSignature, MethodDefinition> **functions** = **new** HashMap<>();  
 **private** Map<MethodSignature, Modificator> **functionsModificator** = **new** HashMap<>();

Он содержит имя класса, поля с модификаторами и методы с модификаторами.

**Результаты**

Вход программы файл с текстом:

class User  
 private id, name  
  
 public function getName()  
 getName = name;  
 end function  
  
 public function setName(\_name as string)  
 name = \_name;  
 end function  
  
 public function getId()  
 getId = id;  
 end function  
  
 public function setId(newId as long)  
 id = newId;  
 end function  
  
 public function New(\_id as long, \_name as string)  
 id = \_id;  
 name = \_name;  
 end function  
  
end class

Мнемоническое описание класса:

---Class---  
User  
---fields---  
0 : id - PRIVATE  
1 : name - PRIVATE  
---functions---  
PUBLIC :

.methodSignature  
MethodSignature{funcName='setId', argCount=1, argsType=[undefined], returnType=undefined}  
.funcs  
.vars\_count  
2  
.consts  
.programm  
0: PUSHVAR 1 0   
1: LOADCLASSVAR 0 0   
2: END   
  
PUBLIC :   
 .methodSignature  
MethodSignature{funcName='getId', argCount=0, argsType=[], returnType=undefined}  
.funcs  
.vars\_count  
1  
.consts  
.programm  
0: PUSHCLASSVAR 0 0   
1: LOADVAR 0 0   
2: END   
  
PUBLIC :   
 .methodSignature  
MethodSignature{funcName='getName', argCount=0, argsType=[], returnType=undefined}  
.funcs  
.vars\_count  
1  
.consts  
.programm  
0: PUSHCLASSVAR 1 0   
1: LOADVAR 0 0   
2: END

PUBLIC :   
 .methodSignature  
MethodSignature{funcName='setName', argCount=1, argsType=[string], returnType=undefined}  
.funcs  
.vars\_count  
2  
.consts  
.programm  
0: PUSHVAR 1 0   
1: LOADCLASSVAR 0 1   
2: END

PUBLIC :

.methodSignature  
MethodSignature{funcName='New', argCount=2, argsType=[long, string], returnType=undefined}  
.funcs  
.vars\_count  
3  
.consts  
.programm  
0: PUSHVAR 1 0   
1: LOADCLASSVAR 0 0   
2: PUSHVAR 2 1   
3: LOADCLASSVAR 1 1   
4: END

**Вывод**

В ходе работы реализована возможность вывода программы в бинарном виде. Добавлена поддержка пользовательских типов данных.