Институт информационных технологий и управления

в технических системах

Кафедра информационных технологий и компьютерных систем

ОТЧЕТ

по лабораторной работе № 11

**«РАЗРАБОТКА КЛАССА-ШАБЛОНА ДЛЯ СОЗДАНИЯ ОБЪЕКТОВ»**

по дисциплине «Программирование. Базовые процедуры обработки информации»

Выполнил студент группы ИВТ/б-11д

Орлов И.В.

Проверил доцент Петров И.И.

Севастополь

2020

**1. ЦЕЛЬ РАБОТЫ**

Целью данной работы является ознакомление с возможностями реализации структур данных, объединяющих данные различных типов (записей) на Java, а также с принципом инкапсуляции данных и методов (подпрограмм) для работы с ними, получение навыков разработки и использования простых классов и созданных на их основе объектов.

2. **ПОСТАНОВКА ЗАДАЧИ**

Разработать программу, создающую объекты заданного класса, выводящую информацию об объектах и осуществляющую операцию сравнения объектов на равенство. Для создания объектов разработать соответствующий класс-шаблон.

Данные для разработки класса «Класс 1» (название реализуемой сущности и ее свойства) приведены в таблице 2.1.

Вариант задания V вычислен по формуле

V = (N%14 != 0) ? N%14 : 14; ,

где N – номер студента в списке группы:

1%14=1;

Таблица 2.1 – Поля (свойства) сущности, которую необходимо описать

в проекте с помощью класса Класс1

|  |  |  |
| --- | --- | --- |
| № | Класс1  (сущность 1) | Поля (private) |
| 1 | Книга | Идентификационный код, название, автор, издательство, год выпуска, количество страниц, тип (1 - учебная, 2 - научная, 3 - художественная), количество экземпляров |

**3. АНАЛИЗ ЗАДАЧИ**

**3.1. Входные данные, выходные данные, функции**

**и структура программы**

Согласно заданию, необходимо разработать java-программу, состоящую из двух классов:

1. «Класс1», который описывает свойства и поведение сущности реального мира, заданной вариантом на лабораторную работу (таблица 2.1);
2. класс ObjDemoLab11, содержащий метод main(), в котором должны создаваться и сравниваться объекты – экземпляры сущности «Класс1».

В методе main () нужно предусмотреть следующие действия:

* + создание объекта класса «Класс 1» с помощью конструктора с параметрами;
  + создание псевдонима для созданного объекта (второй ссылки на объект);
  + создание копии объекта с помощью конструктора без параметров;
  + создание еще одного объекта (с другими значениями полей) с помощью конструктора с параметрами;
  + для всех использованных ссылочных переменных вывод в окно терминала данных об объектах, на которые они ссылаются;
  + вывод в окно терминала результатов сравнения объекта с самим собой, со своей копией, с другим объектом (с другими значениями полей) и с null-ссылкой.

Входные данные программы – значения свойств создаваемых объектов. Выходные данные программы (сведения о созданных объектах и результаты их сравнения) выводятся в окно терминала.

**3.2. Описание класса-шаблона «Класс 1»**

Сущность «Класс 1», согласно заданию, имеет следующие свойства (таблица 2.1): идентификационный код, название, автор, издательство, год выпуска, количество страниц, тип (1 - учебная, 2 - научная, 3 - художественная), количество экземпляров. Значения этих свойств для каждого экземпляра класса должны храниться в соответствующих переменных экземпляра. Следовательно, в классе будет определено пять переменных экземпляра:

id – целочисленная (int);

name – ссылка на объект класса String (символьную строку);

… *дописать для остальных полей*

Все поля (переменные экземпляра) будут объявлены private (скрыты в классе).В правильно записанных Java-программах public-методы определяют, как можно использовать private члены-переменные.

Поведение сущности характеризуют следующие методы:

1. конструктор без параметров;
2. конструктор с параметрами;
3. методы-геттеры (для каждого поля);
4. методы-сеттеры (для каждого поля);
5. метод toString(), выдающий строку описания объекта;
6. метод equals(), сравнивающий объект с другим объектом того же класса (переданным в метод как параметр) на равенство (объекты равны, если ссылки на них равны (один и тот же объект), иначе, если их одноименные поля равны).

Конструктор – это специальный метод класса, который определяет, какие действия должны выполняться при создании объекта данного класса.

Он должен иметь спецификатор public, т.к. конструктор запускается из другого класса. Имя метода-конструктора совпадает с именем класса.

В классе будет два перегруженных конструктора, отличающихся списком параметров: конструктор без параметров (инициализирует поля создаваемого объекта нулевыми значениями) и конструктор с параметрами (инициализирует соответствующие поля объекта переданными аргументами).

Каждый метод-геттер возвращает значение соответствующего поля объекта (параметров не имеет, тип возвращаемого значения соответствует типу поля).

Каждый метод-сеттер позволяет установить значение соответствующего поля (тип параметра совпадает с типом поля, метод не возвращает значения – void).

Метод toString() (без параметров) возвращает ссылку на объект класса String (символьную строку описания объекта). Этот метод **переопределяет** метод суперкласса Object с той же сигнатурой. Если метод toString() не переопределяется в классе, то при обнаружении ссылки на объект методы printf () и println() будут использовать для вывода информации об объекте метод toString(), определенный в суперклассе (будет выводиться значение ссылки – адреса объекта). Если же метод toString() переопределен в классе, то доступ к аналогичному методу суперкласса становится невозможным.

Метод equals(), сравнивающий текущий объект (объект, запустивший метод) с другим объектом того же класса на равенство. Метод возвращает значение типа boolean. В качестве параметра в метод передается ссылка на объект того же класса («Класс1»). Таким образом, метод equals() данного класса **перегружает** одноименный метод суперкласса Object, в который передается в качестве параметра ссылка на объект класса Object. Имеет место перегрузка, а не переопределение, т.к. сигнатуры методов в суперклассе и подклассе - разные. Другими словами, если мы вызовем **от имени объекта класса «Класс1»** метод equals() с аргументом-ссылкой на объект класса «Класс1», то запустится метод, определенный в классе «Класс1». Если же в метод будет передан аргумент-ссылка на объект класса Object, то запустится метод класса Object, который сравнивает лишь адреса объектов.

**4. ТЕСТОВЫЕ ПРИМЕРЫ И РЕЗУЛЬТАТЫ ИХ ОБРАБОТКИ ВРУЧНУЮ**

Для тестирования работы программы, в ней будут созданы объекты со следующими значениями полей:

1)…,

2)…,

… .

Структуры данных, которые должны присутствовать в памяти после создания программой всех объектов, изображены на рисунке 4.1 ***(выполнить подобно рисункам 3.1 – 3.3 в методических указаниях).***

Рисунок 4.1 – Структуры данных для тестового примера

*Далее нужно назвать и обосновать ожидаемые результаты сравнения объектов.*

**5. СТРУКТУРА ПРОЕКТА**

Структура проекта изображена на рисунке 5.1.

Рисунок 5.1 – Структура проекта

**6.ТЕКСТ ПРОГРАММЫ**

Согласно пункту 2.8.1. методических указаний к лабораторным работам, текст программы должен быть отформатирован следующим образом:

1. шрифт: arial, 12 пт, полужирный;
2. абзац:

* выравнивание по левому краю;
* уровень – основной текст;
* отступ слева – 0;
* отступ справа – 0;
* интервал перед – 0;
* интервал после – 0;
* первая строка – нет (отступа или выступа),
* междустрочный интервал – множитель, значение – 1,2;

1. комментарии (обязательно должны присутствовать в тексте программы) выделить курсивом и синим цветом.

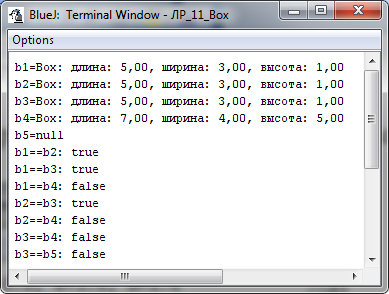
**Для вывода использовать только вызовы метода System.out.printf(). Не использовать операцию конкатенации строк!!!**

**7. СВЕДЕНИЯ ОБ ОТЛАДКЕ ПРОГРАММЫ И ПРОВЕРКЕ**

**ЕЕ РАБОТОСПОСОБНОСТИ**

*В данном пункте сначала следует описать ошибки, которые возникли при отладке программы и способ их устранения.*

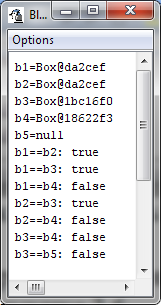
Результаты выполнения программы представлены на рисунке 7.1.



***Вставить свои результаты.***

Рисунок 7.1 – Результаты выполнения программы

Результаты работы программы при отсутствии в классе Box метода toString () (определение метода было закомментировано в программе) приведены на рисунке 7.2.



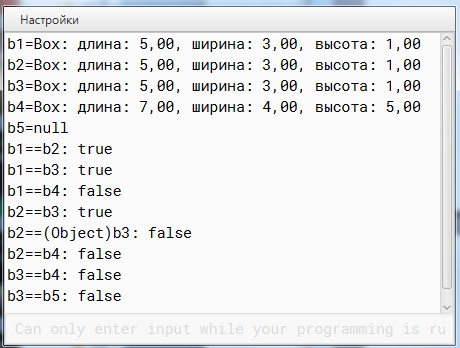
***Вставить свои результаты.***

Рисунок 7.2 – Результаты работы программы

при отсутствии в классе Box метода toString ()

Результаты работы программы при добавлении в метод main() строки

**System.out.printf ("b2==(Object)b3: %s\n", b2.equals((Object)b3));** приведены на рисунке 7.3.

****

***Вставить свои результаты.***

Рисунок 7.3 – Результаты работы программы, осуществляющей

вызовы метода equals() из класса Box (выделено синим) и из

суперкласса Object (выделено красным) для сравнения двух объектов с

одинаковыми значениями полей и разными адресами

***Проанализируйте значения ссылок на рисунке 7.2 и объясните (опишите здесь) результаты сравнения объектов. Объясните (опишите здесь), какое влияние оказывает изменение типа аргумента (явное преобразование в Object) в вызове метода equals(). Объясните (опишите здесь), почему метод toString () является переопределенным, а метод equals () – перегруженным.***

На тестовых примерах программа выдала ожидаемый результат, что позволяет сделать вывод о ее работоспособности.

**ВЫВОД**

В ходе выполнения лабораторной работы были исследованы следующие принципы ООП (и их реализация в Java): принцип инкапсуляции и принцип полиморфизма (параметрический полиморфизм и его частный случай – конструкторский полиморфизм).

Разработана программа, манипулирующая объектами класса «Класс1»: объекты создаются, выводится информация о созданных объектах, объекты сравниваются на равенство. В программе демонстрируется операция присвоения ссылки, в результате которой у объекта появляется еще один псевдоним. В программе также демонстрируется сравнение объекта с самим собой, с его копией, с объектом, отличающимся по значениям полей, и с null-ссылкой. С программой проведены эксперименты, показывающие разницу между переопределенным (override) методом и перегруженным (overload).

Класс «Класс1» является классом, по шаблону которого создаются объекты – экземпляры класса. Поэтому все его члены (переменные и методы) являются экземплярными , а статических переменных и методов в классе нет.

Переменные скрыты в классе с помощью спецификатора доступа private. Доступ к ним осуществляется с помощью public-методов класса. Класс содержит два перегруженных конструктора (конструкторский полиморфизм).

В конструкторе с параметрами проблема сокрытия переменных экземпляра локальными параметрами с теми же именами решается с помощью использования ссылки this (ссылка на текущий объект, т.е. объект, запустивший метод).

Любой класс в Java является наследником класса Object. Метод toString() класса «Класс1» переопределяет одноименный метод суперкласса Object, т.к. имеет такую же сигнатуру. Метод equals() класса «Класс1» перегружает одноименный метод суперкласса Object, т.к. сигнатуры у этих методов разные (отличаются типом параметра – параметрический полиморфизм).

Присутствие в классе «Класс1» метода toString() делает невозможным вызов одноименного метода суперкласса Object от имени объекта подкласса.

Метод equals(), определенный в суперклассе, можно вызвать от имени объекта подкласса, если передать в качестве аргумента ссылку на объект суперкласса.

Для вывода результатов работы программы в методе main () применялся метод System.out.printf(), т.к. он позволяет не использовать операцию конкатенации символьных строк, способствующую появлению в памяти «мусорных» объектов.

***Уважаемые студенты! Обращаю ваше внимание на то, что шаблоны отчетов даются вам для того, чтобы вы научились писать технические отчеты и правильно составлять документацию к программам. Кроме того, я стараюсь вставлять в них важные теоретические сведения. Поэтому настоятельно прошу сначала внимательно прочитать текст шаблона, при необходимости творчески переработать его в соответствии с данными своего варианта и только затем вставлять результаты работы своей программы.***