Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

ОТЧЕТ

по лабораторной работе №3

по курсу «Основы Web-программирования» на тему

«ООП»

Выполнили: студенты группы 21ВП2

Лакеева Софья

Малькова Анастасия

Принял: к.т.н. Такташкин Д.В.

Пенза 2022

# Название

Объектно-ориентированное программирование

# Цель работы

Познакомиться с реализацией объектно-ориентированной парадигмы на языке Java.

# Лабораторное задание

Космические тела

– определить самую далекую от Солнца планету;

– определить планеты, которые ближе к Солнцу, чем Земля;

– упорядочить массив по возрастанию расстояния от Солнца;

– организовать поиск по названию планеты, исправление одного из полей и вывод полной информации о планете после редактирования.

# Листинг

Файл Main.java

*// 21ВП2 Малькова, Лакеева  
// Вариант 8. Лаба 3  
// Космические тела  
//– определить самую далекую от Солнца планету;  
//– определить планеты, которые ближе к Солнцу, чем Земля;  
//– упорядочить массив по возрастанию расстояния от Солнца;  
//– организовать поиск по названию планеты, исправление одного из полей и вывод полной информации о планете после редактирования.***import** java.util.Scanner;  
**import** java.util.ArrayList;  
**import** java.util.regex.Pattern;  
**import static** java.lang.System.\*;  
**public class** Main {  
 **public static void** main(String args[]){  
 Planet Earth = **new** Planet(**"Земля"**, 150, 12.8, 6, 1,365);  
 Planet Mercury = **new** Planet(**"Меркурий"**, 58, 4.9, 0.32, 0, 58);  
 Planet Venus = **new** Planet(**"Венера"**, 108, 12.1, 4.86, 0, 255);  
 Planet Mars = **new** Planet(**"Марс"**, 288, 6.8, 0.61, 2,687);  
 Planet Jupiter = **new** Planet(**"Юпитер"**, 778, 142.6, 1906.98, 16,4380);  
 Planet Saturn = **new** Planet(**"Сатурн"**, 1426, 120.2, 570.9, 17,10950);  
 Planet Uranus = **new** Planet(**"Уран"**, 2869, 49, 87.24, 14,30660);  
 Planet Neptune = **new** Planet(**"Нептун"**, 4496, 50.2, 103.38, 2,60225);  
 Planet Pluto = **new** Planet(**"Плутон"**, 5900, 2.8, 0.1, 1,90885);  
 Star Sun = **new** Star(**"Солнце"**, 0 , 1392, **"5 век до нашей эры"**, 200000, 3.827);  
 Star Sirius = **new** Star(**"Сириус"**, 63000000, 2366, **"1844"**, 400000, 97.2);  
 Star Aldebaran = **new** Star(**"Альдебаран"**, 441000000, 36192, **"17 век"**, 3000000, 1680.1);  
 Earth.arr\_push();  
 Venus.arr\_push();  
 Mercury.arr\_push();  
 Jupiter.arr\_push();  
 Mars.arr\_push();  
 Uranus.arr\_push();  
 Saturn.arr\_push();  
 Pluto.arr\_push();  
 Neptune.arr\_push();  
 Sun.arr\_push();  
 Sirius.arr\_push();  
 Aldebaran.arr\_push();  
 *// Создание объектов с консоли* ***out***.println(**"Создайте объект. Для этого введите название класса (Planet или Star) и \n"** +  
 **"значения полей объекта через запятую и пробел. Поля класса Planet: название(string), расстояние до солнца(int, млн км), \n"** +  
 **"диаметр(double), вес(int, 10^24 кг), число спутников(int), период обращения(int). Поля класса Star: название(string), \n"** +  
 **"расстояние до солнца(int, млн км), диаметр(double), дата открытия(string), вес(int, 10^24), светимость(double).\n"** +  
 **"Создание объектов продолжается, пока вы не введете 0\n"** +  
 **"Пример ввода: Planet, Глизе, 9866, 4.5, 57, 0, 780"**);  
 Scanner in = **new** Scanner(System.***in***);  
 String line = in.nextLine();  
 **while** (!Pattern.*matches*(**"^0$"**, line)){  
 *// Проверка корректности ввода* **while** (!Pattern.*matches*(**"(^(Planet), ([0-9а-яa-zА-ЯA-Z]+ )\*[0-9а-яa-zА-ЯA-Z]+, [0-9]+, "** +  
 **"([1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?), ([1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?), [0-9]+, "** +  
 **"[0-9]+$)|(^(Star), ([0-9а-яa-zА-ЯA-Z]+ )\*[0-9а-яa-zА-ЯA-Z]+, [0-9]+, ([1-9][0-9]\*(\\.[0-9]+)?|"** +  
 **"0(.[0-9]+)?), ([0-9а-я]+ )\*[0-9а-я]+, [0-9]+, ([1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)$)|"** +  
 **"(^((?!Planet|Star)[^, \\.]+)(, [^, ]+)+$)|(^0$)"**, line)){  
 ***out***.println(**"Проверьте порядок полей класса и введите корректные значения полей через запятую и пробел"**);  
 line = in.nextLine();  
 }  
 **if** (line.equals(**"0"**)){  
 **break**;  
 }  
  
 String[] words = line.split(**", "**);  
 **if** (words[0].equals(**"Planet"**)) {  
 String name = words[1];  
 **int** to\_sun = Integer.*parseInt* (words[2]);  
 **double** d = Double.*parseDouble* (words[3]);  
 **double** w = Double.*parseDouble* (words[4]);  
 **int** s = Integer.*parseInt* (words[5]);  
 **int** p = Integer.*parseInt* (words[6]);  
 Planet pl = **new** Planet (name, to\_sun, d, w, s, p);  
 pl.arr\_push();  
 ***out***.println(**"Объект успешно создан! Если хотите завершить создавание объектов - введите 0"**);  
 }  
 **else if** (words[0].equals(**"Star"**)){  
 String name = words[1];  
 **int** to\_sun = Integer.*parseInt* (words[2]);  
 **double** d = Double.*parseDouble* (words[3]);  
 String od = words[4];  
 **double** w = Double.*parseDouble* (words[5]);  
 **double** l= Double.*parseDouble* (words[6]);  
 Star st = **new** Star (name, to\_sun, d, od, w, l);  
 st.arr\_push();  
 ***out***.println(**"Объект успешно создан! Если хотите завершить создавание объектов - введите 0"**);  
 }  
 **else  
 *out***.printf(**"Невозможно создать объект класса %s\n"**, words[0]);  
 line = in.nextLine();  
 }  
 *// Самая далекая от Солнца планета* System.***out***.println(Cosmic\_body.*theFurthest*());  
 *// Планеты, которые ближе к Солнцу, чем Земля* System.***out***.println(Cosmic\_body.*closer\_than\_Earth*());  
 *// Упорядочить массив по возрастанию расстояния от Солнца* ArrayList<Cosmic\_body> s\_arr = Cosmic\_body.*sort\_array*();  
 System.***out***.println(**"Космические тела в порядке возрастания расстояний до Солнца:"**);  
 **for** (Cosmic\_body o : s\_arr)  
 System.***out***.println(o.toString());  
 *// Редактирование поля класса* System.***out***.print(**"Введите имя космического тела: "**);  
 Scanner in1 = **new** Scanner(System.***in***);  
 String name = in1.nextLine();  
 **if** (Cosmic\_body.*find*(name)==**null**){  
 System.***out***.println(**"Объект с таким именем не найден"**);  
 }  
 **else**{  
 Cosmic\_body ob = Cosmic\_body.*find*(name);  
 System.***out***.println(Cosmic\_body.*find*(name));  
 *// Если объект класса Star* **if** (ob.getClass().getName().equals(**"Star"**)){  
 ***out***.println(**"Введите поле класса, которое хотите изменить (имя, расстояние до солнца, диаметр, "** +  
 **"дата открытия, вес, светимость)"**);  
 Scanner in2 = **new** Scanner(System.***in***);  
 String field = in2.nextLine();  
 **while** (!Pattern.*matches*(**"^(имя|расстояние до солнца|диаметр|дата открытия|вес"** +  
 **"|светимость)$"**, field)){  
 ***out***.println(**"Проверьте название поля, которое вы ввели"**);  
 field = in2.nextLine();  
 }  
 ***out***.println(**"Введите новое значение поля"**);  
 Scanner in3 = **new** Scanner(System.***in***);  
 String value = in3.nextLine();  
 **switch** (field){  
 **case** (**"расстояние до солнца"**):  
 **while** (!Pattern.*matches*(**"[0-9]+"**, value)){  
 ***out***.println(**"Расстояние до солнца должно быть целочисленным числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"диаметр"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Диаметр должен быть числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"вес"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Вес должен быть числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"светимость"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Светимость должна быть числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **default**:  
 **break**;  
 }  
 ob.edit(field, value);  
 ***out***.println(**"Редактирование выполнено успешно!\n"**);  
 ***out***.println(ob);  
 }  
 *// Если объект класса Planet* **else if** (ob.getClass().getName().equals(**"Planet"**)){  
 ***out***.println(**"Введите поле класса, которое хотите изменить (имя, расстояние до солнца, диаметр, "** +  
 **"вес, число спутников, период обращения)"**);  
 Scanner in2 = **new** Scanner(System.***in***);  
 String field = in2.nextLine();  
 **while** (!Pattern.*matches*(**"^(имя|расстояние до солнца|диаметр|вес"** +  
 **"|число спутников|период обращения)$"**, field)){  
 ***out***.println(**"Проверьте название поля, которое вы ввели"**);  
 field = in2.nextLine();  
 }  
 ***out***.println(**"Введите новое значение поля"**);  
 Scanner in3 = **new** Scanner(System.***in***);  
 String value = in3.nextLine();  
 **switch** (field){  
 **case** (**"расстояние до солнца"**):  
 **while** (!Pattern.*matches*(**"[0-9]+"**, value)){  
 ***out***.println(**"Расстояние до солнца должно быть целым числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"диаметр"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Диаметр должен быть числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"вес"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Вес должен быть числом"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"число спутников"**):  
 **while** (!Pattern.*matches*(**"[0-9]+"**, value)){  
 ***out***.println(**"Число спутников - целое число"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **case** (**"период обращения"**):  
 **while** (!Pattern.*matches*(**"(^[1-9][0-9]\*(\\.[0-9]+)?|0(.[0-9]+)?)"**, value)){  
 ***out***.println(**"Период обращения - целое число"**);  
 value = in3.nextLine();  
 }  
 **break**;  
 **default**:  
 **break**;  
 }  
 ob.edit(field, value);  
 ***out***.println(ob);  
 }  
 }  
 }  
}  
**abstract class** Cosmic\_body {  
 **private static** ArrayList<Cosmic\_body> *arr* = **new** ArrayList<>();  
 **private static final int *from\_earth\_to\_sun*** = 150;  
 **private** String **name**;  
 **private int distance\_to\_sun**;  
 **private double diameter**;  
 **private double weight**;  
 **public** Cosmic\_body(String name, **int** distance\_to\_sun, **double** diameter, **double** weight) {  
 **this**.**name** = name;  
 **this**.**distance\_to\_sun** = distance\_to\_sun;  
 **this**.**diameter** = diameter;  
 **this**.**weight** = weight;  
 }  
 **public** String getName(){  
 **return name**;  
 }  
 **public int** getDistance\_to\_sun(){  
 **return distance\_to\_sun**;  
 }  
 **public double** getDiameter(){  
 **return diameter**;  
 }  
 **public double** getWeight(){  
 **return weight**;  
 }  
 **public void** setDiameter(**double** d){  
 **this**.**diameter** = d;  
 }  
 **public void** setName(String n){  
 **this**.**name** = n;  
 }  
 **public void** setDistance\_to\_sun(**int** d){  
 **this**.**distance\_to\_sun** = d;  
 }  
 **public void** setWeight(**double** w){  
 **this**.**weight** = w;  
 }  
 **public void** arr\_push(){  
 *arr*.add(**this**);  
 }  
 **public static** String theFurthest(){  
 Cosmic\_body o = *arr*.get(0);  
 **for** (**int** i=0; i<*arr*.size() ;i++){  
 **if** ( (*arr*.get(i) **instanceof** Planet) && (*arr*.get(i).getDistance\_to\_sun()>o.getDistance\_to\_sun()))  
 o = *arr*.get(i);  
 }  
 **return "Самая далекая планета от Солнца:\n"** + o.getName() +**"\n"**;  
 }  
 **public static** String closer\_than\_Earth(){  
 String result=**""**;  
 **for** (**int** i=0; i<*arr*.size(); i++){  
 **if** ((*arr*.get(i) **instanceof** Planet) && (*arr*.get(i).getDistance\_to\_sun()<Cosmic\_body.***from\_earth\_to\_sun***))  
 result+=*arr*.get(i).getName()+**" "**;  
 }  
 **return "Планеты, которые ближе к Солнцу, чем Земля:\n"** + result + **"\n"**;  
 }  
 **static public** ArrayList<Cosmic\_body> sort\_array(){  
 **for** (**int** j= *arr*.size()-1; j>0; j--){  
 **for** (**int** i=0; i<=j-1; i++){  
 **if** (*arr*.get(i).getDistance\_to\_sun()>*arr*.get(i+1).getDistance\_to\_sun()){  
 Cosmic\_body s = *arr*.get(i);  
 *arr*.set(i, *arr*.get(i+1));  
 *arr*.set(i+1, s);  
 }  
 }  
 }  
 **return** *arr*;  
 }  
 **public static** Cosmic\_body find(String n) {  
 **for** (**int** i = 0; i < *arr*.size(); i++) {  
 **if** (*arr*.get(i).getName().equals(n)) {  
 **return** *arr*.get(i);  
 }  
 }  
 **return null**;  
 }  
 *// заменить!* **public abstract void** edit(String f, String v);  
}  
**class** Planet **extends** Cosmic\_body {  
 **private int satellite**;  
 **private int period**;  
 **public** Planet(String name, **int** distance\_to\_sun, **double** diameter, **double** weight, **int** satellite, **int** period){  
 **super**(name, distance\_to\_sun, diameter, weight);  
 **this**.**satellite** = satellite;  
 **this**.**period**=period;  
 }  
 **public int** getSatellite(){  
 **return satellite**;  
 }  
 **public int** getPeriod(){  
 **return period**;  
 }  
 **public void** setSatellite(**int** s){  
 **this**.**satellite** = s;  
 }  
 **public void** setPeriod(**int** p){  
 **this**.**period** = p;  
 }  
 @Override  
 **public** String toString(){  
 **return "Планета: "** + **super**.getName() + **"\nРасстояние до солнца (млн км): "** + **super**.getDistance\_to\_sun() +  
 **"\nДиаметр: "** + **super**.getDiameter() + **"\nПериод обращения: "** + **this**.getPeriod() +  
 **"\nЧисло спутников: "** + **this**.getSatellite() + **"\nМасса (10^24 кг): "** + **super**.getWeight() + **"\n\n"**;  
 }  
 **public void** edit(String f, String v){  
 **switch** (f) {  
 **case** (**"имя"**):  
 **super**.setName(v);  
 **break**;  
 **case** (**"расстояние до солнца"**):  
 **super**.setDistance\_to\_sun(Integer.*parseInt*(v));  
 **break**;  
 **case** (**"диаметр"**):  
 **super**.setDiameter(Double.*parseDouble*(v));  
 **break**;  
 **case** (**"вес"**):  
 **super**.setWeight(Double.*parseDouble*(v));  
 **break**;  
 **case** (**"число спутников"**):  
 **satellite** = Integer.*parseInt*(v);  
 **break**;  
 **case** (**"период обращения"**):  
 **period** = Integer.*parseInt*(v);  
 **break**;  
 **default**:  
 **break**;  
 }  
 }  
}  
**class** Star **extends** Cosmic\_body{  
 **private double luminosity**;  
 **private** String **opening\_date**;  
 **public** Star(String name, **int** distance\_to\_sun, **double** diameter, String opening\_date, **double** weight, **double** luminosity){  
 **super**(name, distance\_to\_sun, diameter, weight);  
 **this**.**luminosity** = luminosity;  
 **this**.**opening\_date** = opening\_date;  
 }  
 **public double** getLuminosity(){  
 **return luminosity**;  
 }  
 **public** String getOpening\_date(){  
 **return opening\_date**;  
 }  
 **public void** setLuminosity(**double** l){  
 **luminosity** = l;  
 }  
 **public void** setOpening\_date(String o){  
 **opening\_date** = o;  
 }  
 @Override  
 **public** String toString(){  
 **return "Звезда: "** + **super**.getName() + **"\nРасстояние до солнца (млн км): "** + **super**.getDistance\_to\_sun() +  
 **"\nДиаметр: "** + **super**.getDiameter() + **"\nМасса (10^24 кг): "** +  
 **super**.getWeight() + **"\nДата открытия: "** + **this**.**opening\_date** + **"\nСветимость (10^26 Вт): "** + **this**.**luminosity** + **"\n\n"**;  
 }  
 **public void** edit(String f, String v){  
 **switch** (f) {  
 **case** (**"имя"**):  
 **super**.setName(v);  
 **break**;  
 **case** (**"расстояние до солнца"**):  
 **super**.setDistance\_to\_sun(Integer.*parseInt*(v));  
 **break**;  
 **case** (**"диаметр"**):  
 **super**.setDiameter(Double.*parseDouble*(v));  
 **break**;  
 **case** (**"дата открытия"**):  
 **opening\_date** = v;  
 **break**;  
 **case** (**"вес"**):  
 **super**.setWeight(Double.*parseDouble*(v));  
 **break**;  
 **case** (**"светимость"**):  
 **luminosity** = Double.*parseDouble*(v);  
 **break**;  
 **default**:  
 **break**;  
 }  
 }  
}

# Пояснительный текст к программе

Класс Cosmic\_body – абстрактный класс, содержащий следующие поля со спецификатором доступа private:

* arr – статический динамический массив, хранящий объекты класса Cosmic\_body;
* from\_earth\_to\_sun **–** статическая константа типа int, равная расстоянию от Земли до Солнца;
* name – переменная типа String, название космического объекта;
* distance\_to\_sun – переменная типа int, расстояние до Солнца;
* diameter – переменная типа double, диаметр космического тела;
* weight – переменная типа double, масса космического тела.

Класс Cosmic\_body содержит следующий нестатические методы:

* public String getName() – возвращает название тела;
* public int getDistance\_to\_sun() – возвращает расстояние до Солнца;
* public double getDiameter() – возвращает диаметр тела;
* public double getWeight() – возвращает массу тела;
* public void setDiameter (double d) – устанавливает новый диаметр;
* public void setName (String n) - устанавливает имя;
* public void setDistance\_to\_sun (int d) – устанавливает расстояние до солнца;
* public void setWeight (double w) устанавливает массу;
* public void arr\_push() – добавляет объект в массив;
* public abstract void edit(String f, String v) – заменяет значение поля объекта.

и статические методы:

* public static String theFurthest() – определяет самую далекую планету от Солнца;
* public static String closer\_than\_Earth() – определяет планеты, которые ближе к Солнцу, чем Земля;
* static public ArrayList<Cosmic\_body> sort\_array() – сортирует массив в порядке возрастания расстояний до Солнца;
* public static Cosmic\_body find(String n) – ищет в массиве объект с названием, указанным в параметре метода;
* public static Cosmic\_body edit(Cosmic\_body o, double d) – изменяет диаметр у найденного тела на новый.

Класс Planet – наследник класса Cosmic\_body, содержит следующие поля со спецификатором доступа private:

* satellite – переменная типа int, число спутников;
* period – переменная типа int, период обращения.

Класс Planet содержит следующие нестатические методы:

* public int getSatellite() - возвращает число спутников;
* public int getPeriod() - возвращает период обращения;
* public void setSatellite (int s) – устанавливает число спутников;
* public void setPeriod (int p) – устанавливает период обращения;
* public void edit(String f, String v) – переопределенный абстрактный метод класса Cosmic\_body;
* public String toString() - переопределенный метод класса Object.

Класс Star – наследник класса Cosmic\_body, содержит следующие поля со спецификатором доступа private:

* luminosity – переменная типа double, светимость;
* opening\_date – переменная типа String, дата открытия.

Класс Star содержит следующие нестатические методы:

* public double getLuminosity() - возвращает светимость;
* public String getOpening\_date() - возвращает дату открытия;
* public void setLuminosity (double l) – устанавливает светимость;
* public void setOpening\_date (String o) – устанавливает дату открытия;
* public String toString()- переопределенный метод класса Object;
* public void edit(String f, String v) – переопределенный абстрактный метод класса Cosmic\_body.

# Результаты работы программы

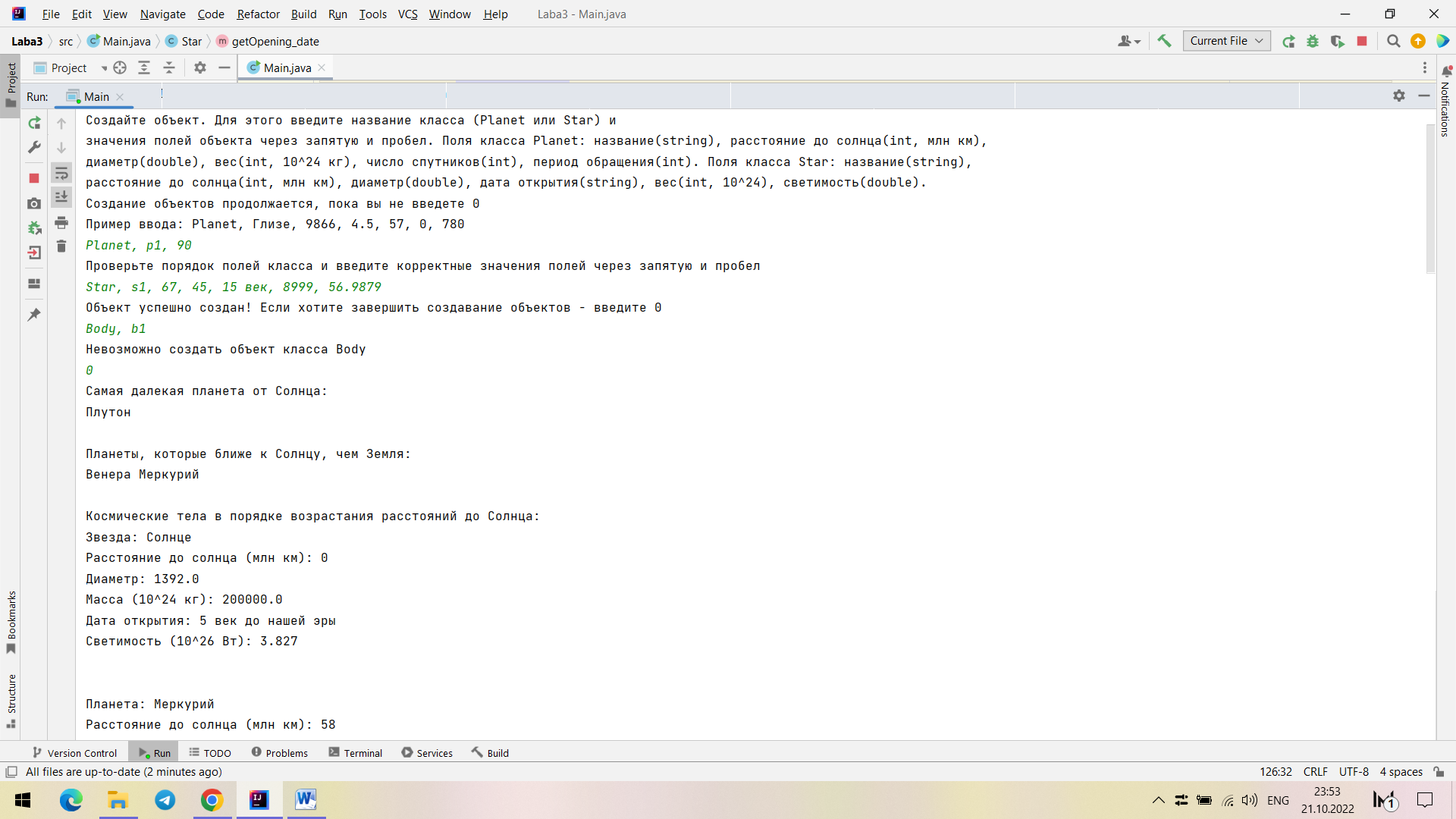
Результаты работы программы показаны на рисунках 1-6.

Рисунок 1 — Результаты работы программы

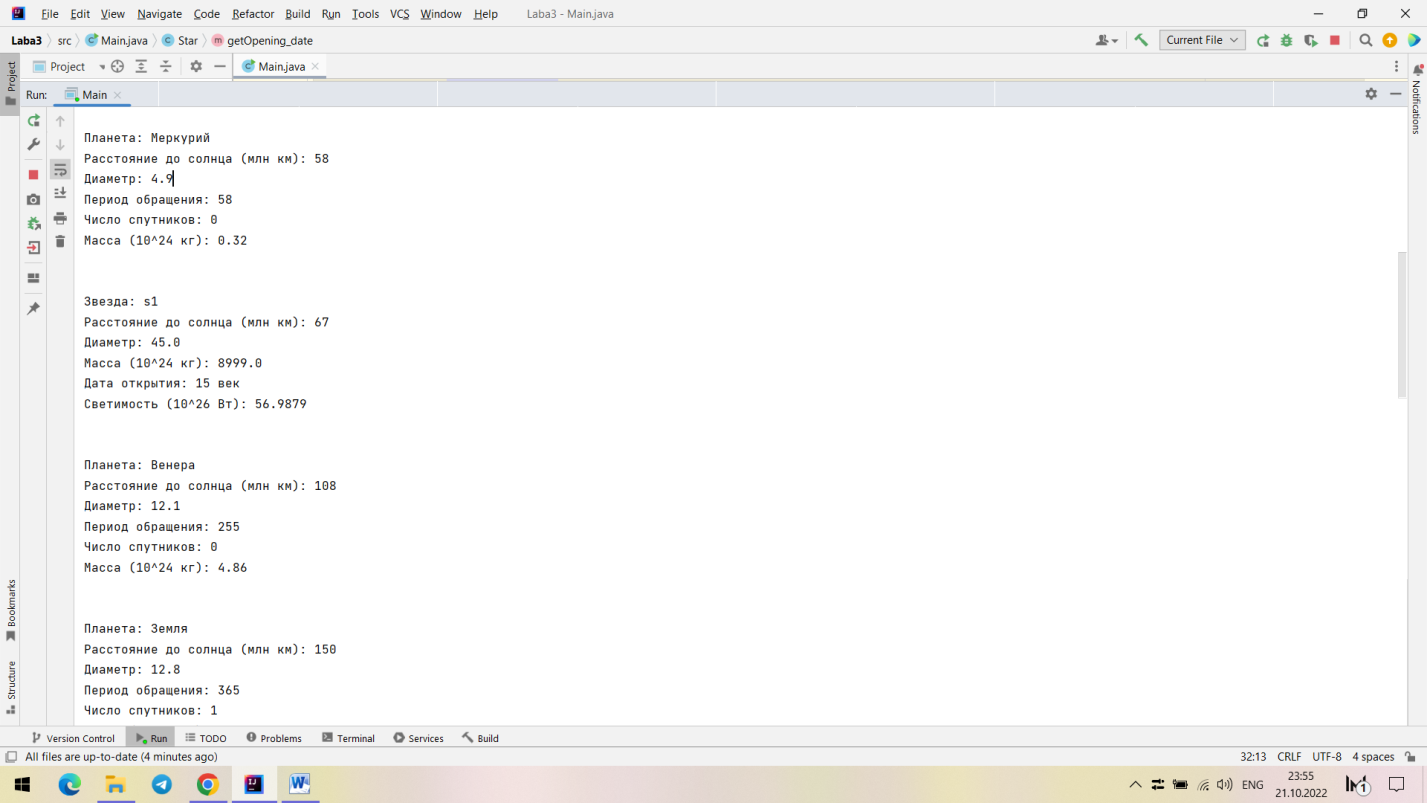


Рисунок 2 — Результаты работы программы

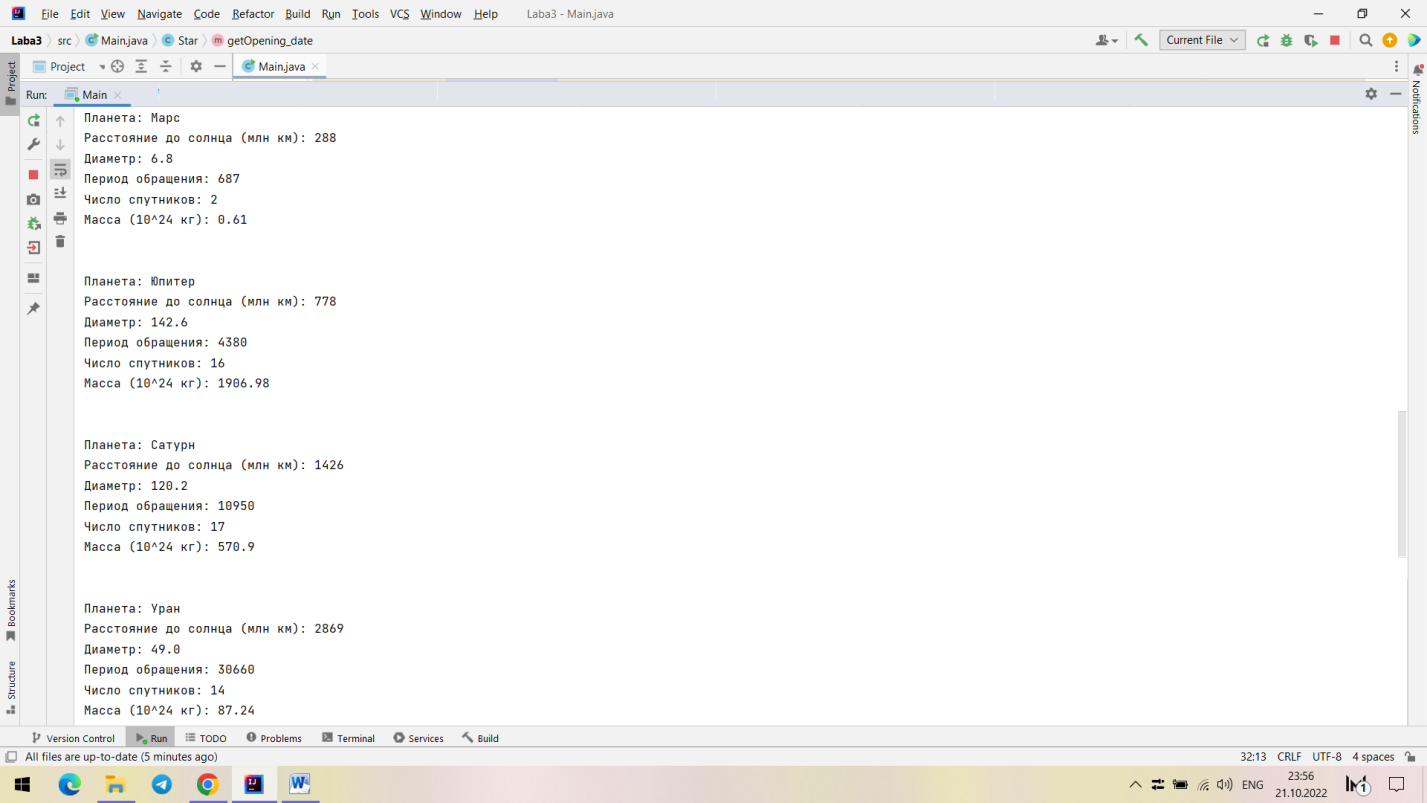
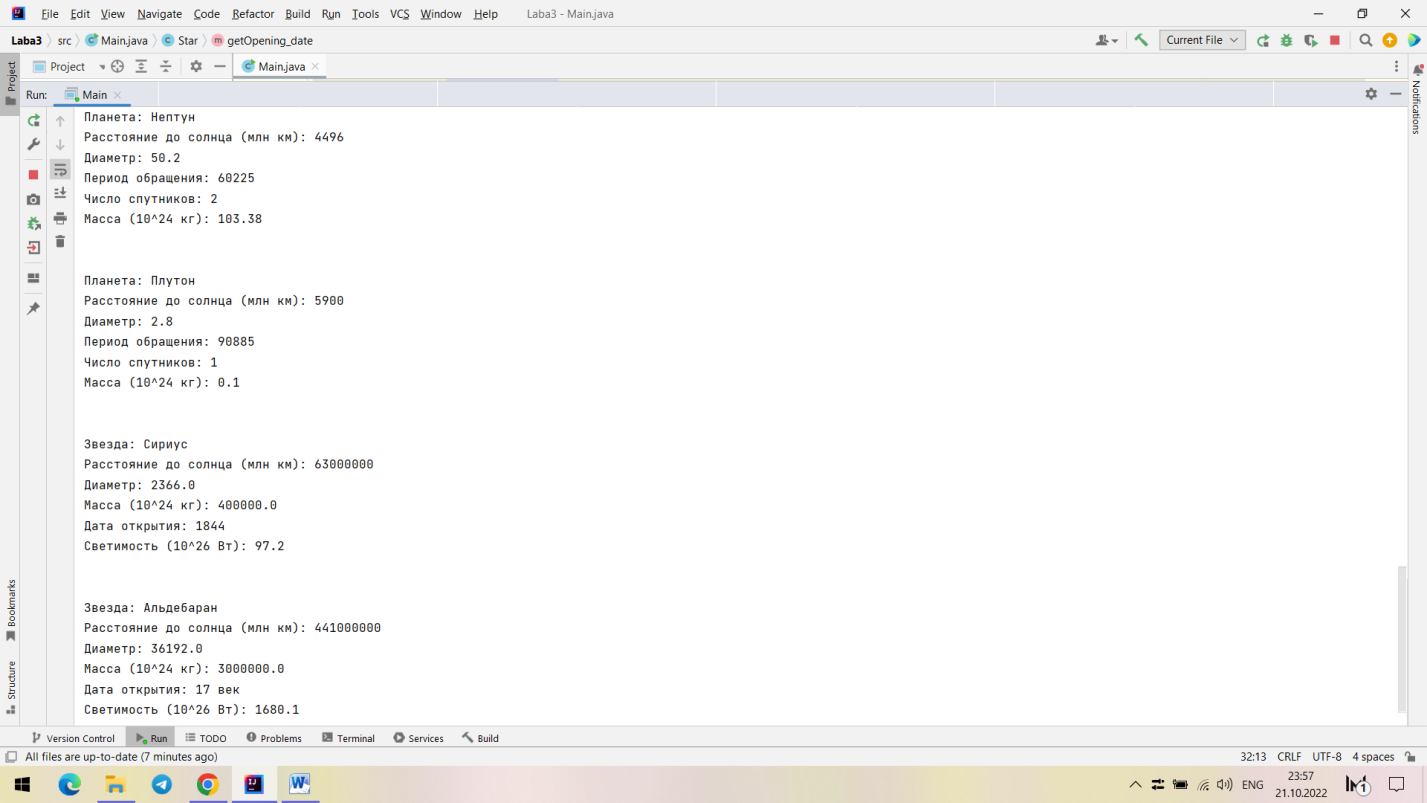


Рисунок 3 — Результаты работы программы



# 

Рисунок 4 — Результаты работы программы

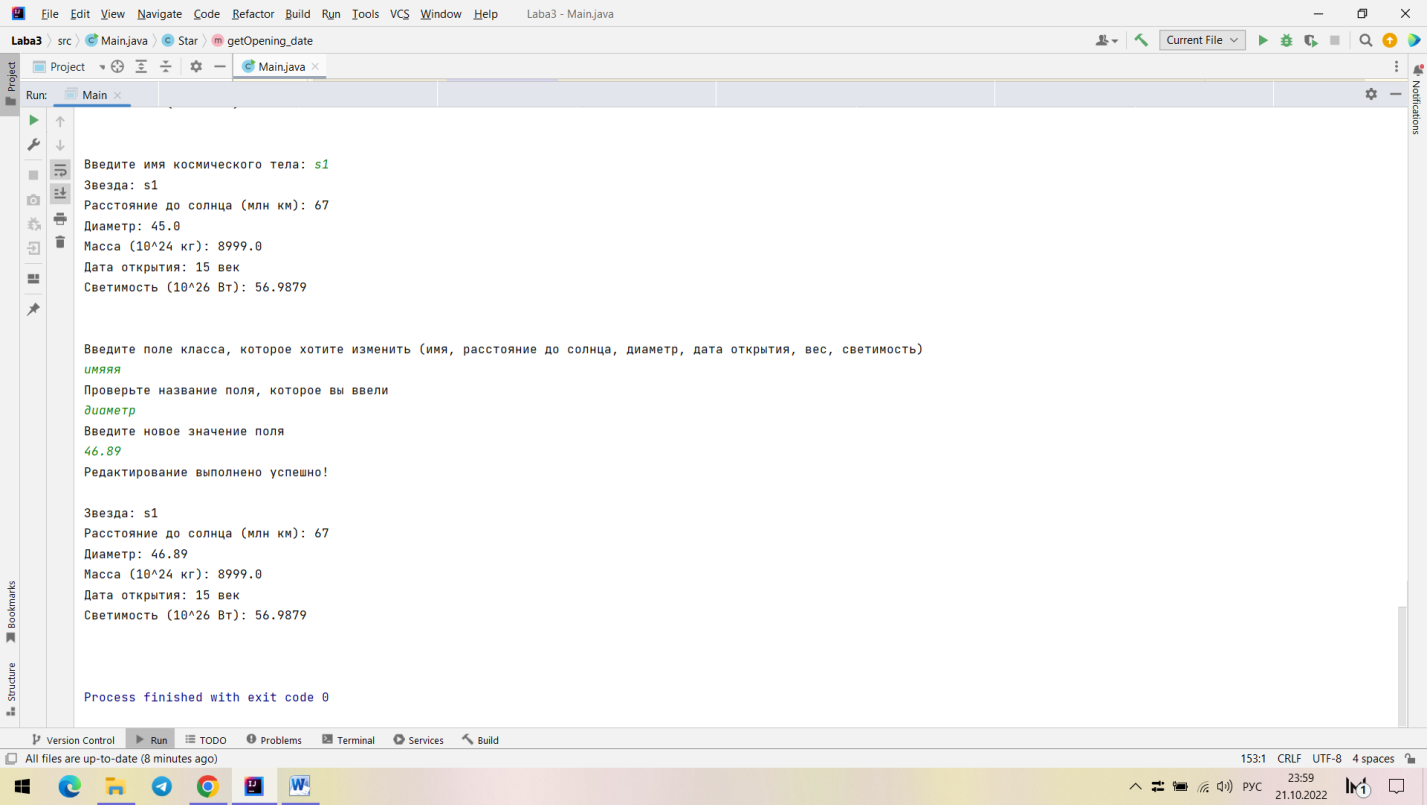


Рисунок 5 — Результаты работы программы

# Выводы

В ходе выполнения лабораторной работы была спроектирована и реализована структура классов предметной области «Космические тела». Была разработана программа, демонстрирующая использование возможностей полученной иерархии классов.