*Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

ДИСЦИПЛИНА Языки программирования для работы с большими данными

**Отчет**

**по лабораторной работе № 4**

**Вариант 5**

Студент гр. ИУ6-21М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Дейкин И. И.\_\_\_

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**ЗАДАНИЕ 1**

*Условие*

Создать класс Календарь с внутренним классом, с помощью объектов которого можно хранить информацию о выходных и праздничных днях.

*Решение*

Класс Календарь с внутренним классом приведен ниже:

|  |
| --- |
| package com.company;  import java.util.ArrayList;  import java.util.List;  public class Calendar {  private List<Holiday> holidays = new ArrayList<Holiday>();  private char[] map = new char[366]; //здесь для каждого дня w - раб день, h - вых  //Изначально все рабочие дни, даже выходные не делал  public Calendar () {  for (int i = 0; i < 366; i++) {  map[i] = 'w';  }  }  public class Holiday {  private String name;  private int date\_strt; //Решил не заморачиваться с датами, сделал интами  private int date\_end;  public Holiday(String s, int d1, int d2) {  this.name = s;  this.date\_end = d2;  this.date\_strt = d1;  }  }  public void AddHoliday(String s, int d1, int d2) {  Holiday h = new Holiday(s, d1, d2);  holidays.add(h);  for (int i = d1; i < d2; i++) {  map[i] = 'h';  }  }  public void Show() {  System.out.println(map);  }  } |

**ЗАДАНИЕ 2**

*Условие*

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов: interface Mobile <- abstract class Siemens Mobile <- class Model.

*Решение*

Описание класса Mobile:

|  |
| --- |
| package com.company;  public interface Mobile {  void Ring();  void Info();  } |

Описание класса SiemensMob:

|  |
| --- |
| package com.company;  public abstract class SiemensMob implements Mobile{  protected String company = "SIEMENS";  String model\_name;  @Override  public void Ring() {  System.out.println("TRR-TRR-TRR");  }  } |

Описание класса ModelT:

|  |
| --- |
| package com.company;  public class ModelT extends SiemensMob{  private String model\_name = "T1242KFDgD";  @Override  public void Info() {  System.out.println(company + ' ' + model\_name);  System.out.println("Has BLANK inside, BLANK outside, Made in China");  }  } |

**ВЫВОД**

В ходе выполнения данной работы было получено представление о внутренних классах, также об абстрактных классах и о интерфейсах.