|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 4**

**Название:** Внутренние классы. Интерфейсы

**Дисциплина:** Языки программирования для работы с большими данными

Студент \_\_\_ИУ6-23М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**А.Д. Зеленский**\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**П. В. Степанов**\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными.

**Вариант 1:**

1. Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.
2. Создать класс Park (парк) с внутренним классом, с помощью объектов которого можно хранить информацию об аттракционах, времени их работы и стоимости.

**Решение:**

Ниже представлен код класса первой программы.

/\*  
8. Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого  
можно хранить информацию об операционной системе, процессоре и оперативной памяти.  
 \*/  
  
  
 class Computer {  
 private int price;  
  
 public class Info{  
 private String os, proc, operat;  
 public void setInfo(String os,String pr,String oper){  
 this.os=os;  
 this.proc=pr;  
 this.operat=oper;  
 }  
  
 public void getInfo(){  
 System.out.println(  
 this.os +" "+  
 this.proc+" "+  
 this.operat+" ;");  
 }  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 System.out.println("Hello world!");  
 }  
}

Ниже представлен код класса второй программы.

/\*  
9. Создать класс Park (парк) с внутренним классом, с помощью объектов которого  
 можно хранить информацию об аттракционах, времени их работы и стоимости.  
 \*/  
  
class Park {  
 private String name;  
  
 public class attrction{  
 private String name, time, price;  
 public void setInfo(String name,String time,String price){  
 this.name=name;  
 this.time=time;  
 this.price=price;  
 }  
  
 public void getInfo(){  
 System.out.println(this.name +" "+  
 this.time+" "+  
 this.price+" ;");  
 }  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 System.out.println("Hello world!");  
 }  
}

**Вариант 2:**

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

1. interface Корабль <- class Грузовой Корабль <- class Танкер..
2. interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.

**Решение:**

Ниже представлен кусок кода первой программы.

/\*  
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:  
interface Корабль <- class Грузовой Корабль <- class Танкер.  
 \*/  
  
interface Ship{  
 void printName();  
}  
  
class CargoShip implements Ship{  
 String name;  
 public CargoShip(){}  
 public CargoShip(String s){  
 this.name=s;  
 }  
  
 @Override  
 public void printName() {  
 System.out.println(this.name);  
 }  
}  
  
class Tancker extends CargoShip{  
 int price;  
 public Tancker(String s){  
 super(s);  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 System.out.println("Hello world!");  
 }  
}

Ниже представлен часть кода второй программы.

/\*  
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:  
interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.  
 \*/  
  
interface Furniture{  
 void printName();  
}  
  
abstract class Closet implements Furniture{  
 String name;  
 public Closet(){}  
 public Closet(String s){  
 this.name=s;  
 }  
 @Override  
 public void printName() {  
 System.out.println(this.name);  
 }  
}  
  
class Bookshelf extends Closet{  
 int price;  
 public Bookshelf(String s){  
 super(s);  
 }  
}  
  
  
public class Main {  
 public static void main(String[] args) {  
 System.out.println("Hello world!");  
 }  
}

**Вывод:** были получены базовые представления о работе языка java с внутренними классами, интерфейсами. Были написаны программы согласно вариантам.