|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |  |

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИППО)

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине «Объектно-ориентированное программирование»

«ООП В JAVA. ПОНЯТИЕ КЛАССА»

ВАРИАНТ 2

Выполнил студент группы ИНБО-01-18 *Трофимов С. М.*

Принял Батанов А.О.

Лабораторная работа выполнена «\_\_\_»\_\_\_\_\_\_\_2019г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2019г.

Москва 2019

**Цель работы**

Получение практических навыков разработки программ, изучение синтаксиса языка Java, освоение основных конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научиться осуществлять стандартный ввод/вывод данных.

**Теоретическое введение**

Язык Java - объектно-ориентированный язык программирования. В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Класс в ООП — это в чистом виде абстрактный тип данных, создаваемый программистом. С этой точки зрения объекты являются значениями данного абстрактного типа, а определение класса задаёт внутреннюю структуру значений и набор операций, которые над этими значениями могут быть выполнены. Желательность иерархии классов (а значит, наследования) вытекает из требований к повторному использованию кода — если несколько классов имеют сходное поведение, нет смысла дублировать их описание, лучше выделить общую часть в общий родительский класс, а в описании самих этих классов оставить только различающиеся элементы.

Необходимость совместного использования объектов разных классов, способных обрабатывать однотипные сообщения, требует 16 поддержки полиморфизма — возможности записывать разные объекты в переменные одного и того же типа. В таких условиях объект, отправляя сообщение, может не знать в точности, к какому классу относится адресат, и одни и те же сообщения, отправленные переменным одного типа, содержащим объекты разных классов, вызовут различную реакцию.

**Выполнение практической работы**

*Задание:*

Создать класс, описывающий тело человека(Human). Для описания каждой части тела создать отдельные классы(Head, Leg, Hand). Описать необходимые свойства и методы для каждого класса. Протестировать работу класса Human.

*Решение:*

Файл Head.java

**package** com.gradess.company;  
  
**public class** Head {  
  
 **private** java.awt.Color **eyesColor**;  
 **private** java.awt.Color **hairColor**;  
  
 **public** Head(java.awt.Color eyesColor, java.awt.Color hairColor) {  
 **this**.**eyesColor** = eyesColor;  
 **this**.**hairColor** = hairColor;  
 }  
  
 **public** java.awt.Color getEyesColor() {  
 **return eyesColor**;  
 }  
  
 **public** java.awt.Color getHairColor() {  
 **return hairColor**;  
 }  
  
 **public void** paintHair(Color hairColor) {  
 **this**.**hairColor** = hairColor;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Цвет глаз: "** + **eyesColor**.toString() +  
 **"\nЦвет волос: "** + **hairColor**;  
 }  
}

Файл Leg.java

**package** com.gradess.company;  
  
**public class** Leg {  
  
 **private int size**;  
  
 **public** Leg(**int** size) {  
 **this**.**size** = size;  
 }  
  
 **public int** getSize() {  
 **return size**;  
 }  
  
 **public void** setSize(**int** size) {  
 **this**.**size** = size;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Размер ноги: "** + **size**;  
 }  
}

Файл Hand.java

**package** com.gradess.company;  
  
**public class** Hand {  
  
 **private float length**;  
  
 **public** Hand(**float** length) {  
 **this**.**length** = length;  
 }  
  
 **public float** getLength() {  
 **return length**;  
 }  
  
 **public void** setLength(**float** length) {  
 **this**.**length** = length;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Длина руки: "** + **length**;  
 }  
}

Файл Color.java

**package** com.gradess.company;  
  
  
  
**public class** Color **extends** java.awt.Color {  
  
 **public** Color(**int** r, **int** g, **int** b) {  
 **super**(r, g, b);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "R = "** + getRed() + **"\tG = "** + getGreen() + **"\tB = "** + getBlue();  
 }  
}

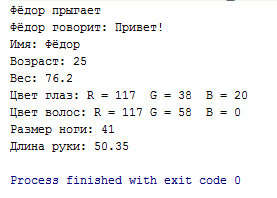
Файл Human.java

**package** com.gradess.company;  
  
**public class** Human {  
  
 **private** String **name**;  
  
 **private int age**;  
  
 **private float weight**;  
  
 **private** Head **head**;  
  
 **private** Leg **leftLeg**;  
 **private** Leg **rightLeg**;  
  
 **private** Hand **leftHand**;  
 **private** Hand **rightHand**;  
  
 **public** Human(String name, **int** age, **float** weight, Head head, Leg leg, Hand hand) {  
 **this**.**name** = name;  
  
 **this**.**age** = age;  
  
 **this**.**weight** = weight;  
  
 **this**.**head** = head;  
  
 **this**.**leftLeg** = leg;  
 **this**.**rightLeg** = leg;  
  
 **this**.**leftHand** = hand;  
 **this**.**rightHand** = hand;  
 }  
  
 **public void** say(String message) {  
 System.***out***.println(**name** + **" говорит: "** + message);  
 }  
  
 **public void** jump() {  
 System.***out***.println(**name** + **" прыгает"**);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Имя: "** + **name** +  
 **"\nВозраст: "** + **age** +  
 **"\nВес: "** + **weight** +  
 **"\n"** + **head** +  
 **"\n"** + **leftLeg** +  
 **"\n"** + **leftHand**;  
 }  
}

Файл Main.java

**package** com.gradess.company;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 Human human = **new** Human(  
 **"Фёдор"**,  
 25,  
 76.2f,  
 **new** Head(**new** Color(117, 38, 20),  
 **new** Color(117, 58, 0)),  
 **new** Leg(41),  
 **new** Hand(50.35f)  
 );  
  
 human.jump();  
 human.say(**"Привет!"**);  
  
 System.***out***.println(human);  
 }  
}

**Результат выполнения программы**



**Выводы по работе**

Я изучил основные концепции объектно-ориентированного программирования, изучил понятие класса и научился создавать классы.

**Используемая литература:**

1. К. Сьерра, Б. Бэйтс. Изучаем Java, 2-е издание.
2. С. Макконнелл. Совершенный код, 3-е издание.