Методические рекомендации по выполнению лабораторных работ по дисциплине «Информационные технологии и программирование»

для студентов направления
09.03.01 Информатика и вычислительная техника

Лабораторная работа №1

Когда мы начинаем изучать язык программирования, первой программой всегда является вывод Hello World. Установим все необходимое и запустим программу на Java, которая выводит приветствие.

Hello World на Java

JavaHelloWorldProgram.java

```
public class JavaHelloWorldProgram {

    public static void main(String args[]){
        System.out.println("Hello World");
    }
}
```

Сохраните приведенную выше программу под именем JavaHelloWorldProgram.java в каталоге для первой лабораторной работы.

Откройте командную строку и перейдите в каталог, в котором сохранен файл программы. Затем выполните следующие команды по порядку.

```
> java JavaHelloWorldProgram.java
```

Программа скомпилируется и запустится, отобразится результат.

Стоит запомнить, что:

- Исходный файл Java может содержать несколько классов, но допускается только один public (это модификатор доступа) класс.
- Имя исходного файла java должно совпадать с именем public класса. Поэтому имя файла нашей программы - JavaHelloWorldProgram.java.
- Когда мы компилируем код, он генерирует байт-код и сохраняет его как расширение Class_Name.class. Если посмотреть на каталог, в котором мы скомпилировали java-файл, то можно заметить новый созданный файл JavaHelloWorldProgram.class
- При выполнении файла класса нам не нужно указывать полное имя файла. Нам нужно использовать только публичное имя класса.
- Когда мы запускаем программу с помощью команды java, она загружает класс в JVM, ищет в классе метод main и запускает его. Синтаксис метода main должен быть таким же,

как указан в примере, иначе программа не будет запущена и выбросит исключение Exception в потоке "main" java.lang.NoSuchMethodError: main.

Задания для выполнения лабораторной работы:

Задание 1 **Создайте программу, которая находит и выводит все простые числа меньше 100.**

Создайте файл с именем Primes.java, в этом файле опишите следующий класс:

```
public class Primes {
        public static void main(String[] args) {
        }
}
```

Внутри созданного класса, после метода main(), опишите метод IsPrime (int n), который определяет, является ли аргумент простым числом или нет. Можно предположить, что входное значение n всегда будет больше 2. Полное описание метода будет выглядеть так:

```
public static boolean isPrime(int n) {
}
```

Данный метод вы можете реализовать по вашему усмотрению, однако простой подход заключается в использовании цикла for. Данный цикл должен перебирать числа, начиная с 2 до (но не включая) n, проверяя существует ли какое-либо значение, делящееся на n без остатка. Для этого можно использовать оператора остатка «%».Если какая-либо переменная полностью делится на аргумент, сработает оператор return false. Если же значение не делится на аргумент без остатка, то это простое число, и сработает оператор return true.

После того, как это будет реализовано, приступайте к заполнению основного метода main() другим циклом, который перебирает числа в диапазоне от 2 до 100 включительно. Необходимо вывести на экран те значения, которые метод IsPrime () посчитал простыми.

После завершения написания программы скомпилируйте и протестируйте её. Убедитесь, что результаты правильные.

Задание 2 **Создайте программу, которая определяет, является ли введенная строка палиндромом.**

Для этой программы, создайте класс с именем Palindrome в файле под названием Palindrome.java. На этот раз вы можете воспользоваться следующим кодом:

```
public class Palindrome {
	public static void main(String[] args) {
		for (int i = 0; i < args.length; i++) {
			String s = args[i];
		}
	}
}
```

Ваша первая задача состоит в том, чтобы создать метод, позволяющий полностью изменить символы в строке. Заголовок метода должен быть следующим:

public static String reverseString(Strings)

Вы можете реализовать этот метод путем создания локальной переменной, которая является пустой строкой, а затем добавлять символы из входной строки в выходные данные, в обратном порядке. Используйте метод length(), который возвращает длину строки, и метод charAt(int index), который возвращает символ по указанному индексу. Индексы начинаются с 0 и увеличиваются на 1.

После того, как вы создали метод reverseString (), создайте еще один метод

public static boolean isPalindrome(String s)

Этот метод должен перевернуть слово s, а затем сравнить с первоначальными данными. Используйте метод equals (Object) для проверки значения равенства. Например:

```
String s1 = "hello";
String s2 = "Hello";
String s3 = "hello";
s1.equals(s2); // Истина
s1.equals(s3); // Ложь
```

Для сравнения строк нельзя использовать оператор «==».

Скомпилируйте и протестируйте программу. Для того чтобы передать аргументы в программу, необходимо указать их при запуске программы через командную строку:

java Palindrome madam racecar apple kayak song noon

Лабораторная работа №2

Основными концепциями ООП являются:

- Абстракция
- Инкапсуляция
- Полиморфизм
- Наследование

Java позволяет описывать объекты (как и любой ООП язык). В данной лабораторной работе необходимо использовать классы, чтобы описать, как эти объекты работают. Вот код для простого класса, который представляет двумерную точку:

Point2d.java

```
public class Point2d {
	/ ** координата X **/
	private double xCoord;
	/ ** координата Y **/
	private double yCoord;
	/ ** Конструктор инициализации **/
	public Point2d (double x, double y) {
		xCoord = x;
		yCoord = y;
	}
	/ ** Конструктор по умолчанию. **/
	public Point2d () {
		this(0, 0);
	}
	/ ** Возвращение координаты X **/
	public double getX () {
		return xCoord;
	}
	/ ** Возвращение координаты Y **/
	public double getY () {
		return yCoord;
	}
	/ ** Установка значения координаты X. **/
	public void setX ( double val) {
		xCoord = val;
	}
	/ ** Установка значения координаты Y. **/
	public void setY ( double val) {
		yCoord = val;
	}
}
```

Экземпляр класса можно также создать, вызвав любой из реализованных конструкторов, например:

```
Point2d myPoint = new Point2d (); //создает точку (0,0)
Point2d myOtherPoint = new Point2d (5,3); //создает точку (5,3)
Point2d aThirdPoint = new Point2d ();
```

При реализации класса трехмерной точки можно использовать описанный выше класс двумерной точки. Для этого необходимо указать что класс Point3d наследует класс Point2d с помощью ключевого слова extends, и далее реализовать функционал, который присущ классу Point3d, но отличается от функционала класса Point2d.

Реализация класса Point3D будет выглядеть следующим образом:

Point3d.java

```
public class Point3d {
    / ** координата Z **/
    private double zCoord;
    / ** Конструктор инициализации **/
    public Point3d (double x, double y, double z){
        super (x, y);
        zCoord=z;
    }
    / ** Конструктор по умолчанию. **/
    public Point3d() {
        this (0,0,0);
    }
    / ** Возвращение координаты Z **/
    public double getZ () {
        return zCoord;
    }
    / ** Установка значения координаты Z. **/
    public void setZ (double val) {
        zCoord = val;
    }
}
```

Весь функционал класса Point2d присутствует в классе Point3d, поэтому необходимо добавить только взаимодействие с 3 координатой. Попробуйте запустить код и создать экземпляры классов Point2d и Point3d, а также установить/получить значения всех координат.

Стоит запомнить, что:

- Важнейшим преимуществом наследования является повторное использование кода, поскольку подклассы наследуют переменные и методы суперкласса.
- Приватные члены суперкласса недоступны для подкласса напрямую, но могут быть косвенно доступна через методы (геттеры и сеттеры).
- Члены суперкласса с модификатором доступа по умолчанию доступны подклассу только в том случае, если они находятся в одном пакете.
- Конструкторы суперкласса не наследуются подклассом.
- Если суперкласс не имеет конструктора по умолчанию, то в подклассе также должен быть определен явный конструктор. В противном случае будет выброшено исключение времени компиляции. В конструкторе подкласса вызов конструктора суперкласса в этом случае обязателен и должен быть первым утверждением в конструкторе подкласса.
- Java не поддерживает множественное наследование, подкласс может расширять только один класс.
- Мы можем создать экземпляр подкласса и затем присвоить его переменной суперкласса, это называется upcasting (повышающее преобразование).
- Когда экземпляр суперкласса присваивается переменной подкласса, это называется downcasting (понижающее преобразование). Нам необходимо явно привести этот экземпляр к подклассу.
- Мы можем переопределить метод суперкласса в классе наследнике. Однако мы всегда должны аннотировать переопределенный метод аннотацией @Override. Компилятор будет знать, что мы переопределяем метод, и если что-то изменится в методе суперкласса, то мы получим ошибку компиляции, а не нежелательные результаты во время выполнения.
- Мы можем вызывать методы суперкласса и обращаться к переменным суперкласса, используя ключевое слово super. Это удобно, когда у нас есть одноименная переменная/метод в подклассе, но мы хотим получить доступ к переменной/методу суперкласса. Это также используется, когда в суперклассе и подклассе определены конструкторы и нам необходимо явно вызвать конструктор суперкласса.

- Мы можем использовать инструкцию instanceof для проверки наследования между объектами.
- В Java мы не можем расширять final классы.
- Если вы не собираетесь использовать суперкласс в коде, т.е. ваш суперкласс является просто базой для сохранения многократно используемого кода, то вы можете сделать его абстрактным классом, чтобы избежать ненужного инстанцирования клиентскими классами. Это также ограничит создание экземпляров базового класса.

Задания для выполнения лабораторной работы:

Задание 1 Создайте иерархию классов в соответствии с вариантом. Ваша иерархия должна содержать:

- абстрактный класс
- 2 уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта)
- демонстрацию реализации всех принципов ООП (абстракция, модификаторы доступа, перегрузка, переопределение)
- наличие конструкторов (в том числе по умолчанию)
- наличие геттеров и сеттеров
- ввод/вывод информации о создаваемых объектах
- предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной, продемонстрируйте работу

Варианты для лабораторной работы №2

1 Базовый класс: Животные Дочерние классы: Кошка, Попугай, Рыбка	2 Базовый класс: Сотрудник Дочерние классы: Администратор, Программист, Менеджер	3 Базовый класс: Человек Дочерние классы: Студент, Преподаватель, Ассистент преподавателя
4 Базовый класс: Транспортное средство Дочерние классы: Легковой автомобиль, Грузовой автомобиль, Мотоцикл	5 Базовый класс Велосипед Дочерние классы: Горный велосипед, Детский велосипед, BMX	6 Базовый класс: Геометрическая фигура Дочерние классы: Шар, Параллелепипед, Цилиндр
7 Базовый класс: Книга Дочерние классы: Аудиокнига, Фильм, Мюзикл	8 Базовый класс: Мебель Дочерние классы: Стол, Стул, Кровать	9 Базовый класс: Монстр Дочерние классы: Гоблин, Русалка, Дракон
10 Базовый класс: Гаджеты Дочерние классы: Часы, Смартфон, Ноутбук	11 Базовый класс: Бытовая техника Дочерние классы: Холодильник, Посудомоечная машина, Пылесос	12 Базовый класс: Приложение Дочерние классы: Социальная сеть, Игра, Погода
13 Базовый класс: Оружие Дочерние классы: Меч, Лук, Волшебная палочка	14 Базовый класс: Заведение Дочерние классы: Кафе, Магазин, Библиотека	15 Базовый класс: Компьютерная периферия Дочерние классы: Клавиатура, Наушники, Графический планшет

Предметная область может быть выбрана другая по согласованию с преподавателем.