Лабораторная работа № 2: Инкапсуляция, полиморфизм, наследование

1. Создайте файл `Cube.java`, демонстрирующий использование различных модификаторов доступа:

```
public class Cube {
    public double width;
    private double height;
    protected double depth;

    public Cube(double width, double height, double depth) {
        this.width = width;
        this.height = height;
        this.depth = depth;
    }

    public void print(){
        System.out.println(this.width + " | " + this.height +
                           " | " + this.depth);
    }
}
```

2. Создайте файл `AccessExample.java`:

```
public class AccessExample {
    public static void main(String[] args) {
        // обращение к полям с разными режимами доступа
        Cube c = new Cube(5,4,3);
        c.print();
        System.out.println(c.width);
        System.out.println(c.depth);
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
5.0 | 4.0 | 3.0
5.0
3.0
```

Модификаторы доступа

В Java предусмотрен контроль пространства имен на уровне пакетов. Пакет в Java представляет собой набор Java-классов и сопутствующих файлов (например, изображений, файлов стилей и т.д.), представленный в виде отдельной директории типа `package`. Использование пакетов позволяет разделить написанные программистом классы конкретной программы и другие классы, находящиеся внутри пакета.

Программист может устанавливать различные режимы доступа к полям и методам класса и области их видимости с помощью следующих ключевых слов (модификаторов доступа):

- `public` – поля и методы, указанные как `public`, являются доступными из любого класса, вне зависимости от того, в каком пакете находится класс;
- `protected` – поля и методы, указанные как `protected`, являются общедоступными внутри текущего пакета и приватными в случае, если класс находится в другом пакете;
- *без модификатора* – поля и методы, указанные без модификатора, являются публичными внутри текущего пакета и приватными за его пределами;
- `private` – поля и методы, указанные как `private`, являются доступными исключительно внутри текущего класса.

	Класс	Пакет	Унаследованный класс	Другое
`public`	Доступен	Доступен	Доступен	Доступен
`protected`	Доступен	Доступен	Доступен	Не доступен
без модификатора	Доступен	Доступен	Не доступен	Не доступен
`private`	Доступен	Не доступен	Не доступен	Не доступен

В приведенном примере, в случае, если класс `Cube` будет помещен в отдельный пакет, доступ к полю `depth` пропадет и возникнет соответствующая ошибка при попытке компиляции и запуска программы.

Поля классов рекомендуется делать приватными (инкапсулировать), создавая геттеры (методы для получения значений полей, например `getWidth()`) и сеттеры (методы для установки значений полей, например `setWidth(double width)`). Таким образом, каждый класс будет обладать набором полей и методы со скрытой реализацией, предоставляя возможность обращения извне исключительно к общедоступным полям с модификатором доступа `public`.

3. Создайте класс `Spy` с полями строкового типа `name` и `realName` с модификаторами доступа `public` и `private` соответственно, приватное поле целочисленного типа `squad`, методами `getSpyInfo()` (возвращает строку с информацией об объекте) с доступом `private` и `print()` с публичным доступом, выводящим информацию об объекте на экран. Для каждого из полей добавьте геттер и сеттер. В отдельном классе проведите испытания для всех полей и методов, выводя информацию на экран.

4. Создайте файл `Triangle.java`:

```
public class Triangle {

    public double side; // сторона

    public Triangle(double side) {
        this.side = side;
    }

    public double area(){
        return(Math.sqrt(3)/4)*Math.pow(this.side,2);
    }
```

```
    public static void checkTriangles(Triangle triangle1,
Triangle triangle2) {
        double area1 = triangle1.area();
        double area2 = triangle2.area();
        if (area1 == area2) {
            System.out.println("Треугольники равны");
        }
        else if (area1 > area2) {
            System.out.println("Первый треугольник больше
второго");
        }
        else {
            System.out.println("Второй треугольник больше
первого");
        }
    }
}
```

5. Создайте файл `StaticExample.java`:

```
public class StaticExample {

    static int result;

    public static void main(String[] args){
        int[] array = {-3, 20, 5, 16, 27};

        // итерация по объектам коллекции
        for (int value: array) {
            // тернарный оператор
            result = (value % 4 == 0) ? 1 : 0;
            System.out.println(result);
        }

        Triangle t1 = new Triangle(3);
        Triangle t2 = new Triangle(4);
        Triangle.checkTriangles(t1,t2);
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
0
1
0
1
0
Второй треугольник больше первого
```

Ключевое слово `static`

Доступ к полям и методам, объявленным с модификатором `static`, можно получить без создания объекта напрямую от соответствующего класса:

```
public class ComedyBook {
    static genre = "comedy";
    …
}
```

Основные особенности модификатора `static`:

- Доступ к полям и методам, объявленным с модификатором `static`, может быть получен только из статических методов;
- Методы, объявленные с модификатором `static`, не могут быть переопределены при наследовании;
- Поля, объявленные с модификатором `static`, инициализируются после загрузки класса в память;
- Для классов, имеющих статические методы, доступен импорт (`import static ...`) данных методов в другие классы, например: `import static java.lang.Math.sqrt` позволит использовать функцию исчисления корня из числа в виде `sqrt(value)` вместо `Math.sqrt(value)`.

6. Создайте класс `StaticContainer` со целочисленным `static`-полем `counter` и публичным `static`-методом `operation()`, реализующим увеличение значения counter на 3 при каждом вызове метода. Метод `operation()` значений не возвращает.

7. Создайте запускаемый класс `StaticCheck`, в главном методе которого реализуйте, используя бесконечный цикл, ожидание получения значения поля `counter` класса `StaticContainer` более 100, после чего завершите цикл и выведите значение `counter` на экран.

8. Создайте файл `Plant.java`:

```
public class Plant {
    private String type;
    private String color;

    public Plant() {}

    public Plant(String type, String color) {
        this.type = type;
        this.color = color;
    }

    public Plant(String type) {
        this.type = type;
    }

    public void print(){
```

```
        System.out.println("type: " + this.type + "; color: " +
this.color);
    }
}
```

9. Создайте файл `OverloadExample.java`:

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

        Plant p1 = new Plant();
        Plant p2 = new Plant("tulip", "red");
        Plant p3 = new Plant("cactus");

        p1.print();
        p2.print();
        p3.print();

    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
type: null; color: null
type: tulip; color: red
type: cactus; color: null
```

Полиморфизм

Методы в Java характеризуются названием, аргументами, возвращаемым значением, режимом доступа. Полиморфизм – способность методов обрабатывать данные разных типов. Программисты могут создавать в классе методы с одинаковым названием, но имеющие при этом различные аргументы. В этом случае метод будет перегружен, при вызове метода из внешнего контекста, компилятор определит, какой именно из методов должен быть вызван по его аргументам. Также следует учитывать, что порядок аргументов при перегрузке методов будет влиять на вызываемый метод.

10. Дополните класс `Plant`, добавив приватные поля `existenceArea` (тип `String`) и `rare` (тип `boolean`). Создайте дополнительные к существующим конструкторы с использованием вновь созданных полей класса. Осуществите проверку в главном методе класса `OverloadExample`.

11. Создайте класс `Ship` со следующими полями:
- `title` (`String`) – название судна;
- `captainName` (`String`) – имя капитана судна;
- `port` (`int`) – номер порта стоянки судна (целое число);
- `type` (`char`) – тип судна (буква латинского алфавита).

Разработайте программу испытаний различных вариантов перегрузки (допустимых и недопустимых) некоторого метода `updateShipInfo()` (`public`, `void`), обновляющего

различные данные о судне, выводя список текущих аргументов метода на экран каждый раз при вызове. На вход метод принимает одно или несколько значений разных полей класса. Вызов методов осуществляйте в отдельном запускаемом классе `ShipTest`.

12. Создайте файл `Furniture.java`:

```
public class Furniture {
    String material;
    double age;

    public Furniture() {}

    public Furniture(String material, double age) {
        this.material = material;
        this.age = age;
    }

    public void print() {
        System.out.println("Furniture: ");
        System.out.println("material:  "  +  this.material  +  ";
age: " + this.age);
    }
}
```

13. Создайте файл `Chair.java`:

```
public class Chair extends Furniture {
    private String color;

    public Chair(String material, double age, String color) {
        super(material, age);
        this.color = color;
    }

    @Override
    public void print() {
        System.out.println("from Furniture: ");
        super.print();
        System.out.println("Chair: ");
        System.out.println("material:  "  +  this.material  +  ";
age: " + this.age +
                           "; color: " + this.color);
    }
}
```

14. Создайте файл `InheritanceExample.java`:

```
public class InheritanceExample {
    public static void main(String[] args){
        Furniture furniture = new Furniture("wood",2.5);
        furniture.print();
        Chair chair = new Chair("aluminum", 1.3, "beige");
        chair.print();
```

```
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
Furniture:
material: wood; age: 2.5
from Furniture:
Furniture:
material: aluminum; age: 1.3
Chair:
material: aluminum; age: 1.3; color: beige
```

Наследование

Наследование в Java реализуется при помощи ключевого слова `extends` в следующем формате:

```
. . . Descendant extends Ancestor { . . . }
```

где `Descendant` – имя дочернего класса, `Ancestor` – имя родительского класса.
Наследуются поля и методы родительских классов.

При наследовании необходимо учитывать следующее:

- поля и методы доступа `private` не наследуются;
- поля и методы доступа `public` должны оставаться `public` в дочерних классах;
- поля и методы доступа `protected` должны быть `public` или `protected` в дочерних классах.

При наследовании можно получить доступ к методам суперкласса (класса-родителя) напрямую из дочернего класса с использованием ключевого слова `super`, например:
`super.printValue();`

Все классы по умолчанию унаследованы от класса `Object`, предоставляющего следующие методы:

- `toString()` – строковое представление объекта класса;
- `equals(Object obj)` – метод сравнения текущего объекта с объектом obj, переданным в качестве аргумента;
- `getClass()` – возвращает имя и класса;
- `notify()` и `notifyAll()` – используются для работы с потоками, ожидающими реакцию объекта;
- `wait()` – переводят поток, обрабатывающий объект, в режим ожидания;
- `hashCode()` – возвращает хэш-код для объекта.

При необходимости указанные методы могут быть переопределены внутри любого дочернего класса.

15. Дополните класс `Chair`. Переопределите метод `toString()`, унаследованный от класса `Object` так, чтобы он возвращал информацию о всех полях объекта класса вместе с их текущими значениями. В классе `InheritanceExample` осуществите проверку переопределенного метода.

16. Используя наследование, реализуйте схему банковских счетов некоторого аккаунта в банке, предполагая наличие как минимум двух счетов для разных типов валюты. Классы должны хранить информацию о номере счета, лимите счета и текущих средствах. В поле строкового типа, объявленном с модификатором `static`, в родительском классе необходимо указать имя владельца счетов.

17. Пусть задана последовательность из `n` положительных чисел `w = (w1, w2, ..., wn)` и `s`. Разработайте программу, которая вычисляет бинарный вектор `x = (x1, x2, ..., xn)`, `(xi = 0 или 1)`, чтобы:

$$s = \sum_{i=1}^{n} x_i w_i$$

Проведите тестирование программы.

18. Пусть `p=43`, а `q=61`. Постройте асимметричную систему шифрования с открытым ключом. Проведите испытание построенной системы.

19. Создайте пакет `systems`, включающий набор классов, описывающих главную систему, и системы, наследуемые от главной, в соответствии с вариантом. При необходимости, создавайте классы, которые наследуются от систем, унаследованных от главной. Часть свойств систем не должны наследоваться другими системами. В каждом из классов (главном и наследуемых) создайте следующие методы:

- ненаследуемые методы для проведения внутренней обработки информации
- ненаследуемые методы для проведения характерных действий по изменению состояния системы
- наследуемые методы для осуществления общедоступных действий с системой:
    - получение (вывод в терминал) информации о свойствах системы
    - редактирование свойств системы
    - общедоступные действия по изменению состояния системы

Создайте запускаемый класс, в котором осуществите проверку всех свойств и методов реализованных классов.

Вариант	**Описание систем**
1	Главная система: «Аудитория» Наследуемые системы: «Компьютерный класс», «Лекционная»
2	Главная система: «Вагон» Наследуемые системы: «Бизнес-класс», «Эконом-класс»
3	Главная система: «Поезд» Наследуемые системы: «Пассажирский поезд», «Товарный поезд»
4	Главная система: «Лампочка» Наследуемые системы: «Светодиодная лампочка», «Люминесцентная лампочка»
5	Главная система: «Самолет» Наследуемые системы: «Пассажирский самолет», «Грузовой самолет»

6	Главная система: «Компьютер» Наследуемые системы: «Ноутбук», «Моноблок»
7	Главная система: «Тренажер» Наследуемые системы: «Велотренажер», «Силовой комплекс»
8	Главная система: «Лампа» Наследуемые системы: «Настольная лампа», «Люстра»
9	Главная система: «Монитор» Наследуемые системы: «Жидкокристаллический монитор», «Кинескопный монитор»
10	Главная система: «Шкаф» Наследуемые системы: «Платяной шкаф», «Книжный шкаф»
11	Главная система: «Олимпиада» Наследуемые системы: «Заочный онлайн-тур», «Очный тур»
12	Главная система: «Транспортное средство» Наследуемые системы: «Сухопутное», «Водное»
13	Главная система: «Телескоп» Наследуемые системы: «Радиотелескоп», «Оптический телескоп»
14	Главная система: «Проигрыватель» Наследуемые системы: «Магнитофон», «Аудиосистема»
15	Главная система: «Водное транспортное средство» Наследуемые системы: «Гидроцикл», «Корабль»
16	Главная система: «Акустическая система» Наследуемые системы: «Однополосная система», «Многополосная система»
17	Главная система: «Сухопутное транспортное средство» Наследуемые системы: «Колесное средство», «Трубопровод»
18	Главная система: «Корабль» Наследуемые системы: «Грузовой корабль», «Пассажирский корабль»
19	Главная система: «Колесное средство» Наследуемые системы: «Автомобиль», «Автобус»
20	Главная система: «Автомобиль» Наследуемые системы: «Грузовой автомобиль», «Легковой автомобиль»
21	Главная система: «Двигатель» Наследуемые системы: «Бензиновый двигатель», «Электродвигатель»
22	Главная система: «Воздушный транспорт» Наследуемые системы: «Авиация», «Воздухоплавание»
23	Главная система: «Авиация» Наследуемые системы: «Самолет», «Вертолет»
24	Главная система: «Досуг» Наследуемые системы: «Творческий досуг», «Культурно-потребительский досуг»
25	Главная система: «Кинотеатр» Наследуемые системы: «2D-кинотеатр», «3D-кинотеатр»
26	Главная система: «Геометрическая фигура» Наследуемые системы: «Плоская», «Объемная»
27	Главная система: «Многоугольник» Наследуемые системы: «Квадрат», «Треугольник»
28	Главная система: «Плоская геометрическая фигура» Наследуемые системы: «Многоугольник», «Кривая»
29	Главная система: «Кривая» Наследуемые системы: «Круг», «Эллипс»

30	Главная система: «База данных» Наследуемые системы: «Реляционная база данных», «Объектная база данных»
31	Главная система: «Образовательное учреждение» (ОУ) Наследуемые системы: «ОУ высшего образования», «ОУ среднего специального образования»
32	Главная система: «Направление» Наследуемые системы: «Техническое направление», «Гуманитарное направление»
33	Главная система: «Подразделение» Наследуемые системы: «Административное подразделение», «Образовательное подразделение»
34	Главная система: «Комедия» (литературный жанр) Наследуемые системы: «Водевиль», «Пародия»
35	Главная система: «Театр» Наследуемые системы: «Опера», «Балет»
36	Главная система: «Средство коммуникации» Наследуемые системы: «Электронная почта», «Мессенджер»
37	Главная система: «Административное подразделение» Наследуемые системы: «Ректорат», «Бухгалтерия»
38	Главная система: «Литературный жанр» Наследуемые системы: «Комедия», «Трагедия»
39	Главная система: «Магазин» Наследуемые системы: «Оптовый магазин», «Розничный магазин»
40	Главная система: «Розничный магазин» Наследуемые системы: «Супермаркет», «Рынок»
41	Главная система: «Репозиторий программного кода» Наследуемые системы: «Публичный репозиторий», «Приватный репозиторий»
42	Главная система: «Гостиница» Наследуемые системы: «Пансионат», «Полупансион»
43	Главная система: «Публикация» Наследуемые системы: «Рецензируемая публикация», «Нерецензируемая публикация»
44	Главная система: «Печатное издание» Наследуемые системы: «Книга», «Журнал»
45	Главная система: «Парк» Наследуемые системы: «Пейзажный парк», «Регулярный парк»
46	Главная система: «Трудоустройство» Наследуемые системы: «Постоянное», «По совместительству»
47	Главная система: «Водоем» Наследуемые системы: «Море», «Озеро»
48	Главная система: «Программное обеспечение» Наследуемые системы: «Системное», «Прикладное»
49	Главная система: «Сотрудник» Наследуемые системы: «Преподаватель», «Административный сотрудник»
50	Главная система: «Тест» Наследуемые системы: «Обучающий тест», «Аттестующий тест»