Лабораторная работа №3

Объектно-ориентированное программирование.

Цель работы: изучение основ объектно-ориентированного программирования в Java.

Основные понятия ООП

Класс – это некоторая модель еще не существующих сущностей (объектов), описывающая их возможное состояние и поведение. Состояние реализуется посредством полей класса, представляющих собой переменные, а поведение посредством методов, представляющих собой функции.

Объект – это отдельный представитель (экземпляр) класса, имеющий конкретное состояние и реализующий своё поведение, полностью определяемое классом.

В Java класс объявляется следующим образом:

```
[public] class ИмяКласса {

    // поле
    [модификаторы] тип_поля имяПоля [= значение];

    // метод
    [модификаторы] тип_возвращаемого_значения ИмяМетода([аргументы]) {
        // тело метода
    }

}
```

Классы, объявленные с модификатором **public** доступны внутри и вне пакета, непубличные классы доступны только внутри пакета (пакетный доступ). Допускается наличие только одного публичного класса в файле.

Конструктор (constructor) – метод, имя которого совпадает с именем класса, предназначенный для инициализации объекта. Вызывается автоматически при создании объекта, не возвращает значения, может принимать на вход параметры, может быть перегружен.

Конструктор по умолчанию – конструктор, не принимающий на вход параметров.

Класс может иметь несколько методов с одинаковым именем, но разным набором аргументов. В этом случае речь идет о **перегрузке методов**. Вызов того или иного метода будет происходить в зависимости от конкретных переменных, переданных в качестве аргументов.

Java поддерживает методы с переменным числом аргументов посредством троеточия:

```
public void ИмяМетода(int... x) {

}
```

Статические элементы класса

Статические поля (static fields) являются общими для всех объектов класса. Объявляются с модификатором static. К таким полям можно обращаться как через объект, так и через имя класса.

```
class Point {
	public static int pointsCreated = 0;
	public int x;
	public int y;

	public Point(int x, int y) {
		pointsCreated++;
		this.x = x;
		this.y = y;
	}
}
```

```
    ...
Point p1 = new Point(0, 0);
Point[] points = {new Point(10, 10), new Point(20, 20), new Point(100, 100)};
System.out.println(Point.pointsCreated); // 4
System.out.println(p1.pointsCreated); // 4
```

Статические методы (static methods) – вызываются для класса, а не для конкретного объекта и могут обращаться напрямую только к статическим элементам класса.

```
public static int getPointsCount() {
	return pointsCreated;
}
```

```
...
System.out.println(Point.getPointsCount()); // 4
```

Наследование

Наследование – механизм, позволяющий создать новый класс на основе существующего. Реализуется посредством ключевого слова **extends**. У класса может быть один родитель и любое количество дочерних классов. Прародителем всех классов Java является класс **java.lang.Object**. Дочернему классу передаются поля и методы родительского класса.

Дочерний класс может обращаться полям и методам, унаследованным от родительского класса, которые объявлены с модификатором **public** или **protected**, или объявлены без

модификатора (**package**), при условии что дочерний класс находится в одном пакете вместе с родительским.

Дочерний класс может иметь свои собственные поля и методы, а также переопределять методы родительского класса.

```
class Point3D extends Point {
	public int z;

	public Point3D(int x, int y, int z) {
		super(x, y);
		this.z = z;
	}
}
```

Для обращения к переопределенным элементам родительского класса из дочернего класса используется ключевое слово **super**.

При создании объекта дочернего класса вначале вызывается конструктор родительского класса, а потом – дочернего. Конструктор родительского класса может быть вызван явно при помощи ключевого слова **super**.

Дочерний класс может иметь поля, совпадающие с названием полей родительского класса. В этом случае, поля, унаследованные от родительского класса, будут скрыты.

Статические методы наследуются, но не переопределяются. Если статический метод дочернего класса совпадает (по имени и параметрам) со статическим методом родительского класса, то метод, унаследованный от родительского класса, скрывается.

Полиморфизм

Полиморфизм – способность дочерних классов переопределять методы родительского класса с тем же названием, реализуя отличающееся поведение. Таким образом, иерархия классов предоставляет одинаковый интерфейс, но разные реализации одного и того же поведения для разных классов.

Например, для класса **Point** мы можем объявить следующий метод:

```
public void info() {
	System.out.println("Точка с координатами (" + this.x + "," + this.y + ")");
}
```

а для класса **Point3D** переопределенный:

```
@Override
public void info() {
	System.out.println("Точка с координатами (" + this.x + "," + this.y + "," +
this.z + ")");
}
```

При переопределении методов нельзя уменьшать их видимость.

Переопределенные методы описываются с директивой @Override.

Инкапсуляция

Инкапсуляция – механизм задания уровня доступа к элементам класса. Инкапсуляция предохраняет данные объекта от нежелательного доступа, позволяя объекту самому управлять доступом к своим данным.

Открытые члены класса составляют внешнюю функциональность, которая доступна другим объектам. Закрытыми (private) обычно объявляются независимые от внешнего функционала члены, а также вспомогательные методы, которые являются лишь деталями реализации и неуниверсальны по своей сути. Благодаря сокрытию реализации класса можно менять внутреннюю логику отдельного класса, не меняя код остальных компонентов системы.

Реализуется следующими модификаторами доступа для элементов класса (полей и методов):

- **public** – элемент класса является общедоступным;
- **private** – элемент доступен только методам класса;
- **protected** – элемент доступен только методам класса и дочерних классов;
- (не задан) – **package** – элемент доступен внутри пакета.

Модификатор	Класс	Пакет	Потомок	Вне класса
public	да	да	да	да
protected	да	да	да	нет
-	да	да	нет	нет
private	да	нет	нет	нет

Абстрактные классы

Абстрактный класс определяет общее поведение для порожденных им классов и предполагает наличие дочерних классов, а также невозможность создать объект этого класса. Объявляется с модификатором **abstract**. Такой класс может содержать или не содержать абстрактные методы.

Класс должен быть объявлен как абстрактный если:

- класс содержит абстрактные методы;
- класс наследуется от абстрактного класса, но не реализует абстрактные методы;
- класс имплементирует интерфейс, но не реализует все методы интерфейса.

Абстрактный метод – это метод, не имеющий реализации. Он объявляется с модификатором **abstract** и должен переопределяется в дочерних классах.

Интерфейсы

Интерфейсы применяются для добавления к классам новых возможностей, которых нет и не может быть в базовых классах. Интерфейсы говорят о том, что класс может делать, но не говорят, как он должен это делать. Интерфейс только гарантирует, что класс выполняет какие-то функции, а как он их выполняет – дело неинтерфейсное.

Описание интерфейса приведено ниже.

```
public interface Computable {
	double compute();
}
```

Все методы, объявленные в интерфейсе, должен реализовывать (имплементировать) класс. Класс, реализующий интерфейс, может иметь свои собственные методы (не объявленные в интерфейсе), может иметь свои собственные поля, должен реализовать все методы интерфейса, либо быть объявлен как абстрактный (abstract).

Интерфейс описывается в отдельном файле *.class.

Пример класса, реализующего интерфейс:

```
class Summator implements Computable {
	private double x;
	private double y;

	public Summator(double x, double y){
		this.x = x;
		this.y = y;
	}

	@Override
	public double compute() {
		return x+y;
	}
}
```

Отличия от абстрактных классов

Абстракный класс	Интерфейс
описывают поведение для иерархии классов	описывают поведение для группы классов, реализующих данный интерфейс
могут обладать состоянием	не могут обладать состоянием
могут реализовывать алгоритмы	(Java SE 7) не могут реализовывать алгоритмы; (Java SE 8) могут реализовывать алгоритмы по умолчанию

объектные ссылки поддерживают динамический полиморфизм	интерфейсные ссылки поддерживают динамический полиморфизм
могут содержать скрытые и защищенные элементы	содержат только публичные элементы
класс может наследоваться только от одного абстрактного класса	класс может реализовывать несколько интерфейсов

Вложенные классы

Вложенный класс (nested class) – класс, объявленный внутри другого класса.

Вложенные классы используются для:

- логической группировки классов
- ограничения доступа к классам
- повышения читабельности и управляемости исходного кода

Вложенные классы:

- статические вложенные классы (static nested classes)
- внутренние классы (inner classes)
- локальные классы (local classes)
- анонимные классы (anonymous classes)

Задание к лабораторной работе:

Построить иерархию классов

№	Задание

1	Создать абстрактный класс «Издание» с методами
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public int getName(); // возвращает название издания
	public Date getDatePublication(); // Дата издания
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Издание → Книга
	→ Журнал
	→ Электронное издание
	
	Написать программу, которая:
	1) Считывает с консоли количество изданий (распознавать количество изданий при

	вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число)) 2) В цикле считывает параметры. Сначала спрашивается тип издания и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (название, автор и т.д.). Понятно, что метод init() разный у разных классов. 3) Считанные объекты добавляются в массив. 4) Ищется самое свежее издание и выводится на экран (вывод через toString())

2	Создать абстрактный класс «Спортивный инвентарь»
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public int getSportType(); // возвращает вид спорта, к которому относится
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Спортивный инвентарь → Мяч → Волейбольный мяч
	→ Теннисный мяч
	→ Ракетка
	→ Метательное копье
	→ Тренажерный инвентарь → Штанга
	→ Гиря
	Написать программу, которая:
	1) Считывает с консоли количество предметов инвентаря (распознавать количество при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число))
	2) В цикле считывает параметры. Сначала спрашивается тип инвентаря и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (вид спорта, к которому относится, характеристики: для мяча — радиус, для гири и штанги — вес и т.д.). Понятно, что метод init() разный у разных классов.
	3) Считанные объекты добавляются в массив.
	4) Ищется весь инвентарь, относящийся к теннису, и выводится на экран (вывод через toString())

3	Создать абстрактный класс «Человек» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public int getAge(); // возвращается возраст человека на текущий момент
	// (полное количество лет)
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	
	Построить иерархию классов:
	Человек → Студент
	→ Школьник
	→ Сотрудник → Преподаватель
	→ Директор

	Написать программу, которая: 1) Считывает с консоли количество человек (распознавать количество человек при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число)) 2) В цикле считывает параметры. Сначала спрашивается тип персоны и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (номер зачетной книжки для студента, номер сертификата для преподавателя и т.д.). Понятно, что метод init() разный у разных классов. 3) Считанные объекты добавляются в массив. 4) Ищется самый младший человек и выводится на экран (вывод через toString())

4	Создать абстрактный класс «Товар» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract int getCost() // возвращает стоимость товара
	public abstract boolean canBuy(int cost); // определяет, можно ли купить товар за имеющуюся // сумму
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Товар → Игрушка
	→ Молочный продукт → Сыр
	→ Сметана
	→ Техника → Камера
	→ Телевизор
	
	Написать программу, которая:
	1) Считывает с консоли количество товаров (распознавать количество товаров при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число))
	2) В цикле считывает параметры. Сначала спрашивается вид товара и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (стоимость, наименование игрушки, разрешение камеры, размер диагонали телевизора и т.д.). Понятно, что метод init() разный у разных классов.
	3) Считанные объекты добавляются в массив
	4) Ищется самый дешевый товар и выводится на экран (вывод через toString())

5	Создать абстрактный класс «Транспортное средство» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract int getLoadCapacity() // возвращает грузоподъемность
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Транспортное средство → Легковой автомобиль
	→ Мотоцикл
	→ Грузовой автомобиль

	Легковая_машина (марка, номер, скорость, грузоподъемность), Мотоцикл (марка, номер, скорость, грузоподъемность, наличие коляски, при этом если коляска отсутствует, то грузоподъемность равна 0), Грузовик (марка, номер, скорость, грузоподъемность, наличие прицепа, при этом если есть прицеп, то грузоподъемность увеличивается в два раза) . Написать программу, которая: 1) Считывает с консоли количество транспортных средств (распознавать количество транспортных средств при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число)) 2) В цикле считывает параметры. Сначала спрашивается вид транспортного средства и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (марка, грузоподъемность и т.д.). Понятно, что метод init() разный у разных классов. 3) Считанные объекты добавляются в массив 4) Ищется транспортное средство с максимальной грузоподъемностью и выводится на экран (вывод через toString())

6	Создать абстрактный класс «Летательный аппарат» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract String getBoardNumber() // возвращает бортовой номер
	public abstract int getMaxSpeed(); // возвращает максимальную скорость
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Летательный аппарат → Вертолет
	→ Аэроплан → Планер
	→ Самолет
	Написать программу, которая:
	1) Считывает с консоли количество летательных аппаратов (распознавать количество летательных аппаратов при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число))
	2) В цикле считывает параметры. Сначала спрашивается вид летательного аппарата и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (бортовой номер, максимальная скорость и т.д.). Понятно, что метод init() разный у разных классов.
	3) Считанные объекты добавляются в массив
	4) Ищется летательный аппарат с наибольшей максимальной скоростью и выводится на экран (вывод через toString())

7	Создать абстрактный класс «Комплектующее» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract int geCost() // возвращает стоимость комплектующего
	public abstract boolean canBuy(int cost); // определяет, можно ли купить товар за
	// имеющуюся сумму
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в

	// Object) Построить иерархию классов: Комплектующее → Процессор → Запоминающее устройство → HDD → CD → Монитор →ЖК → Электронно лучевой Написать программу, которая: 1) Считывает с консоли количество комплектующих (распознавать количество комплектующих при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число)) 2) В цикле считывает параметры. Сначала спрашивается тип комплектующего и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта. Понятно, что метод init() разный у разных классов. 3) Считанные объекты добавляются в массив 4) Ищутся все комплектующие дороже заданной суммы и выводятся на экран (вывод через toString())

8	Создать абстрактный класс «Арифметическая операция» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract int calculate() // возвращает результат выполнения операции
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Арифметическая операция → Унарная операция → Плюс
	→ Минус
	→ Бинарная операция →Сложение
	→ Вычитание
	→Умножение
	→ Деление
	Написать программу, которая:
	1) Считывает с консоли количество операция (распознавать количество операция при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число))
	2) В цикле считывает параметры. Сначала спрашивается тип операции и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта (операнды). Понятно, что метод init() разный у разных классов.
	3) Считанные объекты добавляются в массив
	4) Ищется операция с максимальным результатом (calculate()) и выводится на экран(вывод через toString())

9	Создать абстрактный класс «Бытовая техника» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract String getCost() // возвращает стоимость
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в

	// Object) Построить иерархию классов: Бытовая техника →Газовая плита →Электронная техника → Микроволновка → Стиральная машинка Написать программу, которая: 1) Считывает с консоли количество техники (распознавать количество техники при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число)) 2) В цикле считывает параметры. Сначала спрашивается тип техники и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта. Понятно, что метод init() разный у разных классов. 3) Считанные объекты добавляются в массив 4) Ищется бытовая техника с наибольшей стоимостью и выводится на экран (вывод через toString())

10	Создать абстрактный класс «Фигура» с методами:
	public abstract void init(Scanner scanner) // считывание параметров с консоли
	public abstract String getPerimetr() // возвращает периметр
	public String toString() // возвращается состояние объекта в виде строки
	// (определяется только в наследниках, т.к. определен в
	// Object)
	Построить иерархию классов:
	Фигура →Треугольник
	→ Окружность →
	→ Квадрат
	Написать программу, которая:
	1) Считывает с консоли количество фигур (распознавать количество фигур при вводе пользователем таких строк: вап4явап, аа4а555 (в этом случае надо распознать первое число))
	2) В цикле считывает параметры. Сначала спрашивается тип фигуры и создается объект нужного класса. Затем у объекта вызывается метод init() и вводятся характеристики объекта. Понятно, что метод init() разный у разных классов.
	3) Считанные объекты добавляются в массив
	4) Ищется фигура с наибольшим периметром и выводится на экран (вывод через toString())

Каждый отчет должен содержать:

1. Заголовок лабораторной работы (название и цель работы).
2. Фамилия, инициалы и группа студента.
3. Задание к лабораторной работе.
4. Краткие теоретические сведения.
5. Описание алгоритмов, функций, примененных решений.
6. Результаты выполнения программ.
7. Исходный код программ.
8. Выводы о проделанной работе.