**ЛАБОРАТОРНАЯ РАБОТА №3 (часть 1)**

**НАСЛЕДОВАНИЕ. ПЕРЕОПРЕДЕЛЕНИЕ МЕТОДОВ**

Цель: изучить механизм наследования в ООП, научится использовать перегруженные и переопределённые методы.

**Понятие и определение наследования**

Наследование является одним из ключевых свойств системы, которое позволяет описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Используя наследование, можно создать главный класс, который определяет свойства, общие для набора связанных элементов.

Дерево

Дуб

Береза

Сосна

Ель

Рис. 3.1 Графическая интерпретация наследовании

Класс Дерево обладает общими свойствами для подклассов Береза, Дуб, Сосна, Ель (рис. 3.1). В жизни такими свойствами выступают наличие ствола, ветвей, корней и т.д.

Класс, от которого происходит наследование, называется суперклассом (или родительским классом). Класс, который выполняет наследование, называется подклассом (или дочерним классом). На рис. 3.1 суперклассом выступает класс Дерево, а подклассами – Береза, Дуб, Сосна и Ель.

Класс (подкласс) может наследовать переменные и методы другого класса (суперкласса) при помощи ключевого слова extends. В таком случае определение подкласса будет выглядеть следующим образом:

Class SubClass extends SuperClass {

// тело класса SubClass;

}

Стоит обратить внимание на то, что в Java невозможно множественное наследование, т.е. можно расширить только один класс.

**Использование ключевых слов this и super**

Как уже было изучено ранее, ключевое слово this используется, если необходимо обратиться к элементам собственного класса. Оно предопределяет ссылку на объект текущего класса.

Пример 3.1 Использование ключевого слова this

class Rectangle {

private double width;

private double height;

Rectangle (double a, double b) {

this.width=a;

this.height=b;

}

}

Подкласс имеет доступ ко всем открытым переменным и методам (кроме private) родительского класса. Доступ осуществляется через ключевое слово super. Оно используется для вызова конструктора суперкласса (super()) или для доступа к члену суперкласса (super.).

При вызове конструктора слово super() всегда должно стоять на первом месте.

super(список\_параметров); /\* вызов конструктора суперкласса с передачей параметров или без нее\*/

super.i = n; /\* обращение к атрибуту суперкласса \*/

super.methodName(); /\* вызов метода суперкласса \*/

Пример 3.2 Использование ключевого слова super

class Point{

protected int x, int y;

public Point (int x1, int y1){

x=x1;

y=y1;

}

}

class Circle extends Point{

int R;

public Circle(int x1, x2, R1){

super (x1,x2);

R=R1;

}

}

**Переопределение методов**

Если два метода с одинаковыми именами и возвращаемыми значениями находятся в одном классе, то списки их параметров должны отличаться. Такие методы являются перегруженными (overloading). Если метод подкласса совпадает с методом суперкласса (родительского класса), то метод подкласса (дочернего класса) переопределяет (overriding) метод суперкласса.

Переопределение методов является основой концепции динамического связывания, реализующей полиморфизм. Когда переопределенный метод вызывается через ссылку суперкласса, Java определяет, какую версию метода вызвать, основываясь на типе объекта, на который имеется ссылка. Таким образом, тип объекта определяет версию метода на этапе выполнения.

В следующем примере переопределяемый метод show() находится в двух классах Fruits и Apple.

Пример 3.3 Переопределение метода

public class Fruits {

private String name;

private int price;

public Fruits(int p, String str) {

name = str;

price = p;

}

public float getPrice(){

return price;

}

public String getName(){

return name;

}

void show(){

System.out.println("название: " + name+ ", цена: " + price);

}

}

public class Apple extends Fruits{

private boolean seed;

public Apple(int p, String str, boolean f) {

super(p, str);

seed = f;

}

void show(){

System.out.println("название:" + getName()+ ", цена: " +getPrice() + ", наличие семечек:" + seed);

}

}

public class ShowInformation {

public static void main(String[] args) {

Fruits fruit1 = new Fruits(5000, "апельсин");

Fruits fruit2 = new Apple(3000, "яблоко", true);

fruit1.show();

fruit2.show();

}

}

Результат выполнения программы:

название: апельсин, цена: 5000

название:яблоко, цена: 3000.0, наличие семечек: true

Объект fruit1 создается при помощи вызова конструктора класса Fruits, и, соответственно, при вызове метода show() вызывается версия метода из класса Fruits. При создании объекта fruit2 ссылка типа Fruits инициализируется объектом типа Apple. При таком способе инициализации ссылка на суперкласс получает доступ к методам, переопределенным в подклассе.

Стоит заметить, что статические методы могут быть переопределены в подклассе, но не могут быть полиморфными, так как их вызов не затрагивает объекты.

При объявлении совпадающих по сигнатуре полей в суперклассе и подклассах их значения не переопределяются и никак не пересекаются, то есть существуют в одном объекте независимо друг от друга. В этом случае задача извлечения требуемого значения определенного поля, принадлежащего классу в цепочке наследования, ложится на программиста.

Пример 3.4 Совпадение полей в суперклассе и подклассах

public class A {

int x = 1, y = 2;

public A() {

y = getX();

System.out.println("в классе A после вызова" + " getX() x=" + x + " y=" + y);

}

public int getX(){

System.out.println("в классе A");

return x;

}

}

public class B extends A{

int x = 3, y = 4;

public B() {

System.out.println("в классе B x=" + x + " y=" + y);

}

public int getX(){

System.out.println("в классе B");

return x;

}

}

public class DemoAB {

public static void main (String[] args) {

A objA = new B();

B objB = new B();

System.out.println("Вывод значения х из объекта А: " + objA.x);

System.out.println("Вывод значения х из объекта В: " + objB.x);

}

}

Результат выполнения программы:

в классе B

в классе A после вызова getX() x=1 y=0

в классе B x=3 y=4

в классе B

в классе A после вызова getX() x=1 y=0

в классе B x=3 y=4

Вывод значения х из объекта А: 1

Вывод значения х из объекта В: 3

# Пример выполнения индивидуального задания

Создайте класс «Геометрическая фигура», содержащий следующую информацию: координаты центра на плоскости. Предусмотреть get методы и метод класса «расчет площади». Этот метод должны переопределять производные классы.

Создайте класс «Треугольник» производный от «Геометрическая фигура» содержит дополнительную информацию: сторона1, сторона2, сторона 3, тип.

Создайте класс «Прямоугольник» производный от «Геометрическая фигура» содержит дополнительную информацию: ширина, высота.

Создайте класс «Окружность» производный от «Геометрическая фигура» содержит дополнительную информацию: радиус.

Пример 3.5 Пример выполнения индивидуального задания:

public class GeomFig { //создание суперкласса

protected int x,y;

float square;

public GeomFig(int x1,int y1){

x=x1;

y=y1;

}

public GeomFig(){

x=5;

y=5;

}

public int GetDatax(){

return x;

}

public int GetDatay(){

return y;

}

public float FindSquare () { // метод расчета площади, который будет //переопределяться в подклассах

square=x\*y;

return square;

}

}

public class Triangle extends GeomFig { //класс Triangle наследует класс GeomFig

private int a,b,c;

String type;

public Triangle (int x, int y,int a, int b, int c, String type) {

super(x,y); //вызов конструктора с параметрами из суперкласса

this.a=a;

this.b=b;

this.c=c;

this.type=type;

}

public Triangle (int x, int y) {

super(x,y);// вызов конструктора с параметрами из суперкласса

a=3;

b=5;

c=5;

type="isosceles";

}

public Triangle () {

super();//вызов конструктора без параметров из суперкласса

a=7;

b=7;

c=7;

type="equilateral";

}

public int GetDataa(){

return a;

}

public int GetDatab(){

return b;

}

public int GetDatac(){

return c;

}

public String Gettype(){

return type;

}

public float FindSquare (){ //переопределение метода расчета площади

square=(float) Math.sqrt(((a+b+c)/2)\*(((a+b+c)/2)-a)\*(((a+b+c)/2)-b)\*(((a+b+c)/2)-c));

return square;

}

}

public class Rectangle extends GeomFig{ // класс Rectangle наследует класс GeomFig

private int width, height;

public Rectangle(int x, int y, int w, int h) {

super(x,y); //вызов конструктора с параметрами из суперкласса

width=w;

height=h;

}

public Rectangle (int x, int y, int w){

super(x,y);// вызов конструктора с параметрами из суперкласса

width=w;

height=8;

}

public Rectangle() {

super();//вызов конструктора без параметров из суперкласса

width=7;

height=6;

}

public int GetDatawidth(){

return width;

}

public int GetDataheight(){

return height;

}

public float FindSquare (){

square=width\*height;

return square;

}

}

public class Circle extends GeomFig {

int rad;

public Circle(int x, int y, int r){

super(x,y);// вызов конструктора с параметрами из суперкласса

rad=r;

}

public Circle(int x, int y){

super(x,y);// вызов конструктора с параметрами из суперкласса

rad=3;

}

public Circle(){

super();//вызов конструктора без параметров из суперкласса

rad=2;

}

public int GetDataRad(){

return rad;

}

public float FindSquare () { //переопределение метода расчета площади

square=(float) (3.14\*rad\*rad);

return square;

}

}

public class WriterInfo { // класс для вывода информации о классах на экран

public static void main(String[] args) {

Rectangle rect1= new Rectangle(1,2,3,4);

System.out.println("Rectangle 1 - Center:("+rect1.GetDatax()+", "+rect1.GetDatay()+") Weidth:"+rect1.GetDatawidth()+" Height:"+rect1.GetDataheight()+" Square:"+rect1.FindSquare()+"; ");

Rectangle rect2=new Rectangle(7,6,5);

System.out.println("Rectangle 2 - Center:("+rect2.GetDatax()+", "+rect2.GetDatay()+") Weidth:"+rect2.GetDatawidth()+" Height:"+rect2.GetDataheight()+" Square:"+rect2.FindSquare()+"; ");

Rectangle rect3=new Rectangle();

System.out.println("Rectangle 3 - Center:("+rect3.GetDatax()+", "+rect3.GetDatay()+") Weidth:"+rect3.GetDatawidth()+" Height:"+rect3.GetDataheight()+" Square:"+rect3.FindSquare()+"; ");

Circle circ1=new Circle(2,2,4);

System.out.println("Circle 1 - Center:("+circ1.GetDatax()+", "+circ1.GetDatay()+") Radius:"+circ1.GetDataRad()+" Square:"+circ1.FindSquare()+"; ");

Circle circ2=new Circle(4,1);

System.out.println("Circle 2 - Center:("+circ2.GetDatax()+", "+circ2.GetDatay()+") Radius:"+circ2.GetDataRad()+" Square:"+circ2.FindSquare()+"; ");

Circle circ3=new Circle();

System.out.println("Circle 3 - Center:("+circ3.GetDatax()+", "+circ3.GetDatay()+") Radius:"+circ3.GetDataRad()+" Square:"+circ3.FindSquare()+"; ");

Triangle trin1=new Triangle (4,5,6,7,8,"scalene");

System.out.println("Triangle 1 - Center:("+trin1.GetDatax()+", "+trin1.GetDatay()+") Side 1:"+trin1.GetDataa()+" Side 2:"+trin1.GetDatab()+" Side 3:"+trin1.GetDatac()+" Type:"+trin1.Gettype()+" Square:"+trin1.FindSquare()+"; ");

Triangle trin2=new Triangle(15,6);

System.out.println("Triangle 2 - Center:("+trin2.GetDatax()+", "+trin2.GetDatay()+") Side 1:"+trin2.GetDataa()+" Side 2:"+trin2.GetDatab()+" Side 3:"+trin2.GetDatac()+" Type:"+trin2.Gettype()+" Square:"+trin2.FindSquare()+"; ");

Triangle trin3=new Triangle();

System.out.println("Triangle 3 - Center:("+trin3.GetDatax()+", "+trin3.GetDatay()+") Side 1:"+trin3.GetDataa()+" Side 2:"+trin3.GetDatab()+" Side 3:"+trin3.GetDatac()+" Type:"+trin3.Gettype()+" Square:"+trin3.FindSquare()+"; ");

}

}

Результат выполнения программы:

Rectangle 1 - Center:(1, 2) Weidth:3 Height:4 Square:12.0;

Rectangle 2 - Center:(7, 6) Weidth:5 Height:8 Square:40.0;

Rectangle 3 - Center:(5, 5) Weidth:7 Height:6 Square:42.0;

Circle 1 - Center:(2, 2) Radius:4 Square:50.24;

Circle 2 - Center:(4, 1) Radius:3 Square:28.26;

Circle 3 - Center:(5, 5) Radius:2 Square:12.56;

Triangle 1 - Center:(4, 5) Side 1:6 Side 2:7 Side 3:8 Type:scalene Square:15.491934;

Triangle 2 - Center:(15, 6) Side 1:3 Side 2:5 Side 3:5 Type:isosceles Square:3.2426405;

Triangle 3 - Center:(5, 5) Side 1:7 Side 2:7 Side 3:7 Type:equilateral Square:16.431677;

**Задания для самостоятельного выполнения**

Задание. Создать соответствующие классы. Определите в каждом классе три перегруженных конструктора: с полным набором параметров, с частью параметров и конструктор по умолчанию. Предусмотреть переопределение методов базового класса в производных классах. Создайте дополнительный класс WriterInfo, для вывода информации о классах.

Вариант 1. Создайте класс «Служащий», содержащий следующую информацию: ФИО, пол, дата рождения, стаж работы, зарплата. Предусмотреть get методы и метод класса «выплачиваемые налоги». Этот метод должны переопределять производные классы.

Создайте класс «Рабочий» производный от «Служащий» содержит дополнительную информацию: номер цеха, разряд.

Создайте класс «Начальник цеха» производный от «Служащий» содержит дополнительную информацию: название цеха, количество подчиненных.

Создайте класс «Директор завода» производный от «Служащий» содержит дополнительную информацию: срок избрания.

Вариант 2. Создайте класс «Аудитория», содержащий следующую информацию: номер, этаж, площадь. Предусмотреть get методы и метод класса «расчет вместимости». Этот метод должны переопределять производные классы.

Создайте класс «Лекционная аудитория» производный от «Аудитория» содержит дополнительную информацию: количество рядов, количество посадочных мест в ряду, наличие проектора.

Создайте класс «Учебный класс» производный от «Аудитория» содержит дополнительную информацию: количество парт, наличие компьютера.

Создайте класс «Лаборатория» производный от «Аудитория» содержит дополнительную информацию: количество компьютеров.

Вариант 3. Создайте класс «Книга», который содержит следующую информацию: издательство, автор и количество страниц. Предусмотреть get методы. Метод класса «Название», который потом будут переопределять производные классы.

Создайте класс «Художественная книга» производный от «Книга» содержит дополнительную информацию: жанр.

Создайте класс «Научная книга» производный от «Книга» содержит дополнительную информацию: тематика, сложность.

Вариант 3. Создайте класс «Одежда», который содержит следующую информацию: производитель, цена и стиль. Предусмотреть get методы. Метод класса «Бренд», который потом будут переопределять производные классы.

Создайте класс «Куртка» производный от «Одежда» содержит дополнительную информацию: сезонность, наличие капюшона.

Создайте класс «Кофта» производный от «Одежда» содержит дополнительную информацию: длина рукава, процент содержания шерсти.

Создайте класс «Брюки» производный от «Одежда» содержит дополнительную информацию: расклешенность.

Вариант 5. Создайте класс «Посуда», который содержит следующую информацию: производитель, цена и стиль. Предусмотреть get методы. Метод класса «Наименование», который потом будут переопределять производные классы.

Создайте класс «Кастрюля» производный от «Посуда» содержит дополнительную информацию: объем и наличие крышки.

Создайте класс «Сковорода» производный от «Посуда» содержит дополнительную информацию: длина ручки, покрытие.

Создайте класс «Тарелка» производный от «Посуда» содержит дополнительную информацию: диаметр.

Вариант 6. Создайте класс «Человек», содержащую следующую информацию: ФИО, пол, возраст. Предусмотреть get методы. Метод класса «Имя», который потом будут переопределять производные классы.

Создайте класс «Ребенок» производный от «Человек» содержит дополнительную информацию: номер детского сада.

Создайте класс «Подросток» производный от «Человек» содержит дополнительную информацию: номер школы, успеваемость в школе.

Создайте класс «Родитель» производный от «Человек» содержит дополнительную информацию: количество детей, место работы.

Вариант 7. Автомобильный транспорт описывается классом «Автомобиль», который содержит следующую информацию: марка автомобиля, тип двигателя (карбюраторный или дизельный), мощность двигателя, масса автомобиля, пробег в км. Предусмотреть get методы. Метод класса «Марка», который потом будут переопределять производные классы.

Создайте класс «Легковой автомобиль» производный от «Автомобиля» содержит дополнительную информацию: расход топлива на 100 км, время разгона до 100 км/ч в секундах.

Создайте класс «Грузовой автомобиль» производный от «Автомобиля» содержит дополнительную информацию: грузоподъемность в тоннах, тип кузова (открытый или закрытый).

Создайте класс «Автобус» производный от «Легкового автомобиля» содержит дополнительную информацию: количество мест.

Вариант 8. Создайте класс «Обувь», содержащую следующую информацию: размер, производитель, цвет. Предусмотреть get методы. Метод класса «Бренд», который потом будут переопределять производные классы.

Создайте класс «Сапоги» производный от «Обувь», который содержит дополнительную информацию: размер каблука, тип (демисезонные или зимние).

Создайте класс «Босоножки» производный от «Обувь» содержит дополнительную информацию: открытость (сильно или нет).

Создайте класс «Кроссовки» производный от «Обувь» содержит дополнительную информацию: тип\_подошвы, фирма\_производитель.

Вариант 9. Создайте класс «Учащийся», содержащий следующую информацию: ФИО, пол, возраст. Предусмотреть get методы. Метод класса «Учебное\_заведение» который потом будут переопределять производные классы.

Создайте класс «Школьник» производный от «Учащийся» содержит дополнительную информацию: номер класса и букву.

Создайте класс «Студент», производный от «Учащийся», который содержит дополнительную информацию: факультет, специальность, номер группы.

Создайте класс «Аспирант» производный от «Учащийся» содержит дополнительную информацию: название научной работы, научный руководитель.

Вариант 10. Класс описывает мебель и содержит следующие данные-члены: производитель, цена и материал. Предусмотреть get методы. Метод класса «Производитель», который потом будут переопределять производные классы.

Создайте класс «Шкаф» производный от «Мебель» содержит дополнительную информацию: ширину, глубину и высоту.

Создайте класс «Стол» производный от «Мебель» содержит дополнительную информацию: количество ящиков, площадь рабочей поверхности.

Создайте класс «Стул» производный от «Мебель» содержит дополнительную информацию: возможность регулировать положение спинки.

Вариант 11. Создайте класс «Персонаж», содержащий следующую информацию: имя, здоровье, сила и ловкость. Предусмотреть get методы. Метод класса «Атаковать», который потом будут переопределять производные классы.

Создайте класс «Воин» производный от «Персонаж» содержит дополнительную информацию: вид оружия и форма щита.

Создайте класс «Монстр» производный от «Персонаж», который содержит дополнительную информацию: длина зубов и когтей, количество\_лап.

Создайте класс «Маг» производный от «Персонаж» содержит дополнительную информацию: вид\_магии.

Вариант 12. Создайте класс «Фильм», содержащий следующую информацию: режиссер, продолжительность и аудитория. Предусмотреть get методы. Метод класса «Название», который потом будут переопределять производные классы.

Создайте класс «Мультфильм» производный от «Фильм» содержит дополнительную информацию: художник и стиль\_графики.

Создайте класс «Мюзикл» производный от «Фильм», который содержит дополнительную информацию: музыкальный\_инструмент, количество\_песен.

Создайте класс «Исторический» производный от «Фильм» содержит дополнительную информацию: исторический\_период, степень\_соответствия\_событиям.

Вариант 13. Создайте класс «Молочный\_продукт», содержащий следующую информацию: срок\_годности, количество\_белка и жирность. Предусмотреть get методы. Метод класса «Производитель», который потом будут переопределять производные классы.

Создайте класс «Йогурт» производный от «Молочный\_продукт» содержит дополнительную информацию: вкусовая\_добавка, объем\_тары, термизированность.

Создайте класс «Творог» производный от «Молочный\_продукт», который содержит дополнительную информацию: масса, зернистость.

Создайте класс «Кефир» производный от «Молочный\_продукт» содержит дополнительную информацию: объем\_тары и количество\_бифидобактерий.

Вариант 13. Создайте класс «Игра», содержащий следующую информацию: год\_выпуска, рейтинг и ОС. Предусмотреть get методы. Метод класса «Название», который потом будут переопределять производные классы.

Создайте класс «Стратегия» производный от «Игра», который содержит дополнительную информацию: максимальное\_количество\_юнитов, колчество\_карт и возможность\_создать\_карту.

Создайте класс «Шутер» производный от «Игра», который содержит дополнительную информацию: количество\_уровней и вид\_оружия.

Создайте класс «Квест» производный от «Игра» содержит дополнительную информацию: возраст.

Вариант 15. Создайте класс «Сайт», содержащий следующую информацию: название, наличие рекламы. Предусмотреть get методы. Метод класса «Популярность», который потом будут переопределять производные классы.

Создайте класс «Социальная\_сеть» производный от «Сайт», который содержит дополнительную информацию: количество\_пользователей, наличие\_приложений.

Создайте класс «Информационный» производный от «Сайт», который содержит дополнительную информацию: количество\_новостей, частота\_обновлений.

Создайте класс «Образовательный» производный от «Сайт» содержит дополнительную информацию: тематика.

**Контрольные вопросы**

1. Что собой представляет механизм наследования?
2. Дайте определение терминам «суперкласс» и «подкласс».
3. С помощью какого ключевого слова предопределяется ссылка на объект текущего класса?
4. В каких случаях используется super(), а в каких super.?
5. В чем разница между перегруженными и переопределенными методами?
6. Каким образом реализуется полиморфизм?