Практическая работа №4. Наследование в Джава. Абстрактные классы.

Цель: познакомиться на практике с реализацией принципа ООП Наследование в Джава и освоить на практике работу с наследованием от абстрактных классов.

Теоретические сведения об абстрактных классах и наследовании

Наследование один из основных принципов разработки объектно-ориентированных программ. В терминологии Джава базовый класс, от которого производится наследование называется суперкласс. Производные классы, называются подклассы наследуют все компоненты родителей (поля и методы) кроме конструкторов и статических компонентов. Обратится к методам родительского класса можно, используя служебное слово super. Основное преимущество наследования — это возможность повторного использования кода. Наследование поддерживает концепцию «возможности повторного использования», т.е. когда мы хотим создать новый класс и уже существует класс, который включает в себя часть кода, который нам нужен, мы можем получить наш новый класс из уже существующего класса. Таким образом, мы повторно используем поля и методы уже существующего класса. Для организации наследования в Джава, используется ключевое слово – extends, что в переводе на русский означает расширяет. Таким образом создавая производный класс с помощью наследования, мы расширяем родительский класс как новыми свойствами – поля данных, так добавляем к нему новое поведение – методы класса. Пример на листинге 4.1

Листинг 4.1 – Пример наследования

```
// базовый класс велосипед
class Bicycle {
public int gear; // поле
public int speed; // поле

// конструктор класса
public Bicycle(int gear, int speed){
this.gear = gear;
```

```
this.speed = speed;
}
//метод класса
public void applyBrake(int decrement){
speed -= decrement;
}
//метод класса
public void speedUp(int increment){
speed += increment;
}
// метод toString()чтобы печатать объекты Bicycle
public String toString(){
return ("No of gears are " + gear + "\n"
+ "speed of bicycle is " + speed);
}
} // end of class

// производный класс горный велосипед
class MountainBike extends Bicycle {
public int seatHeight; //новое поле произв. класса
//конструктор производного класса
public MountainBike(int gear, int speed,
int startHeight){
// здесь вызов конструктора класса родителя
super(gear, speed);
seatHeight = startHeight;
}
// новый метод производного класса
public void setHeight(int newValue)
    {
        seatHeight = newValue;
    }=
// переопределенный метод toString()класса Bicycle
@Override public String toString(){
return (super.toString() + "\nseat height is "
+ seatHeight);
```

```
  }
}
// класс тестер Main
public class Main {
public static void main(String args[]){
// создаем объект родительского класса
Bicycle bl = new Bicycle(5,200);
System.out.println(bl.toString());

// создаем объект дочернего класса
MountainBike mb = new MountainBike(3, 100, 25);
System.out.println(mb.toString());
    }
}
```

Абстрактные классы

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы при определении помечаются ключевым словом abstract.

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип. Он состоит только из объявления и не имеет тела:

```
abstract void yourMethod();
```

По сути, мы создаём шаблон метода. Например, можно создать абстрактный метод для вычисления площади фигуры в абстрактном классе Фигура. А все другие производные классы от главного класса могут уже реализовать свой код для готового метода. Ведь площадь у прямоугольника и треугольника вычисляется по разным алгоритмам и универсального метода не существует.

Если вы объявляете класс, производный от абстрактного класса, но хотитеиметь возможность создания объектов нового типа, вам придётся предоставить определения для всех абстрактных методов базового класса. Если этого несделать, производный класс тоже останется абстрактным, и компилятор заставитпометить новый класс ключевым словом **abstract**.

Абстрактный класс не может содержать какие-либо объекты, а также абстрактные конструкторы и абстрактные статические методы. Любой подкласс абстрактного класса должен либо реализовать все абстрактные методы суперкласса, либо сам быть объявлен абстрактным. Пример

приведен на листинге 4.2

Листинг 4.2 – Пример абстрактного класса Swim

```
public abstract class Swim {
// абстрактный метод
abstract void swim()
// абстрактный класс может содержать и обычный метод
void run() {
System.out.println("Куда идешь?");
}
}
//создаем производный класс Swimmer
class Swimmer extends Swim {
….
}
```

Задания на практическую работу №4

1. Необходимо реализовать простейший класс Shape (Фигура). Добавьте метод класса getType() (тип фигуры, возвращает строку тип String название фигуры). С помощью наследования создайте дочерние классы Circle, Rectangle и Square. (из предыдущей практической работы). Также реализуйте во всех классах методы getArea()(возвращает площадь фигуры), getPerimeter() взвращает периметр фигуры). Переопределите в дочерних класс методы класса родителя toString(), getArea(), getPerimeter() и getType(). Создать класс-тестер для вывода информации об объекте и продемонстрировать вызов методов использую родительскую ссылку. Объяснить работу программы.

2. Создайте класс Phone, который содержит переменные number, model и weight.

1)Создайте три экземпляра этого класса. 2) Выведите на консоль значения их переменных. 3) Добавить в класс Phone методы: receiveCall, имеет один параметр – имя звонящего. 4)Выводит на консоль сообщение "Звонит {name}". 5)Метод getNumber – возвращает номер телефона. 6) Вызвать эти методы для каждого из объектов. 7) Добавить конструктор в класс Phone, который принимает на вход три параметра для инициализации переменных класса - number, model и weight. 8)Добавить конструктор, который принимает на вход два параметра для инициализации переменных

класса - number, model. 9)Добавить конструктор без параметров. 10)Вызвать из конструктора с тремя параметрами конструктор с двумя. 11)Добавьте перегруженный метод. receiveCall, который принимает два параметра - имя звонящего и номер телефона звонящего. 12)Вызвать этот метод. 13)Создать метод sendMessage с аргументами переменной длины. Данный метод принимает на вход номера телефонов, которым будет отправлено сообщение. 14)Метод выводит на консоль номера этих телефонов.

3. Создать класс Person, который содержит: а) поля fullName, age. б) методы move() и talk(), в которых просто вывести на консоль сообщение -"Такой-то Person говорит". в) Добавьте два конструктора - Person() и Person(fullName, age). Создайте два объекта этого класса. Один объект инициализируется конструктором Person(), другой - Person(fullName, age).

4. Создать класс Матрица. Класс должен иметь следующие поля: 1) двумерный массив вещественных чисел; 2) количество строк и столбцов в матрице. Класс должен иметь следующие методы: 1) сложение с другой матрицей; 2) умножение на число; 3) вывод на печать; 4) умножение матриц - по желанию.

5. Класс «Читатели библиотеки». Определить класс Reader, хранящий такую информацию о пользователе библиотеки: ФИО, номер читательского билета, факультет, дата рождения, телефон. Методы takeBook(), returnBook(). Разработать программу, в которой создается массив объектов данного класса. Перегрузить методы takeBook(), returnBook(): - takeBook, который будет принимать количество взятых книг. Выводит на консоль сообщение "Петров В. В. взял 3 книги". - takeBook, который будет принимать переменное количество названий книг. Выводит на консоль сообщение "Петров В. В. взял книги: Приключения, Словарь, Энциклопедия". - takeBook, который будет принимать переменное количество объектов класса Book (создать новый класс, содержащий имя и автора книги). Выводит на консоль сообщение “Петров В. В. взял книги: Приключения, Словарь, Энциклопедия”. Аналогичным образом перегрузить метод returnBook(). Выводит на консоль сообщение “Петров В. В. вернул книги: Приключения, Словарь, Энциклопедия”. Или “Петров В. В. вернул 3 книги”.

6. Создайте пример наследования, реализуйте класс Employer и класс Manager. Manager отличается от Employer наличием дополнительных выплат от продаж а) Класс Employer содержит переменные: String

firstName, lastName и поле income для заработной платы. Класс Manager также имеет поле double averageSum содержащую среднюю суммы дополнительных выплат за продажи. б) Создать переменную типа Employer, которая ссылается на объект типа Manager. в) Создать метод getIncome() для класса Employer, который возвращает заработную плату. Если средняя количество отработанных дней, то сумма дохода умножается на 12. Переопределить этот метод в классе Manager и добавить к доходу сумму с продаж. г) Создать массив типа Employer содержащий объекты класса Employer и Manager. Вызвать метод getIncome() для каждого элемента массива.

7. Создать суперкласс Учащийся и подклассы Школьник и Студент. Создать массив объектов суперкласса и заполнить этот массив объектами. Показать отдельно студентов и школьников.

Задания на абстрактные классы

8. Перепишите суперкласс Shape из задания 1, сделайте его абстрактным и наследуйте подклассы, так как это представлено на UML диаграмме на рис. 4.1 Circle, Rectangle и Square.

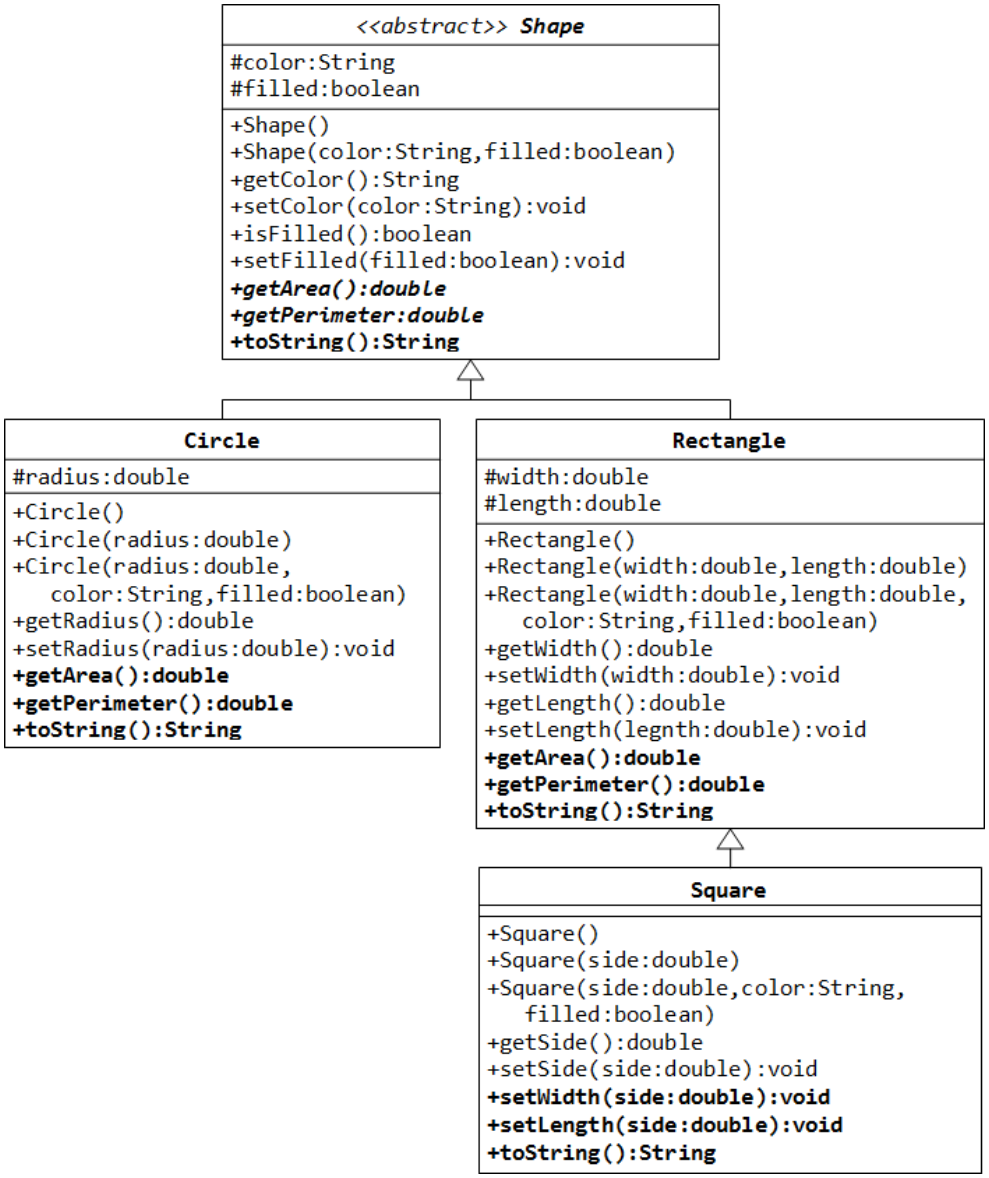

Рисунок 4.1 Диаграмма суперкласса Shape.

Замечания. В этом задании, класс Shape определяется как абстрактный класс, которыйсодержит:

- Два поля или переменные класса, объявлены с модификатором ***protected*** color (тип String) и filled (тип boolean). Такие защищенные переменные могут быть доступны в подклассах и классах в одном пакете. Они обозначаются со знаком “#” на диаграмме классов в нотации языка UML.
- Методы геттеры и сеттеры для всех переменных экземпляра класса,и метод toString().
- Два абстрактных метода getArea() и getPerimeter() выделены курсивом в диаграмме класса).

В подклассах Circle (круг) и Rectangle (прямоугольник) должны

переопределяться абстрактные методы getArea() и getPerimeter(), чтобы обеспечить их надлежащее выполнение для конкретных экземпляров типа подкласс. Также необходимо для каждого подкласса переопределить toString() .

9. Создать абстрактный класс, описывающий сущность мебель. С помощью наследования реализовать различные виды мебели. Также создать класс FurnitureShop, моделирующий магазин мебели. Протестировать работу классов.

10. Создать абстрактный класс, описывающий Транспортное средство и подклассы Автомобиль, Самолет, Поезд, Корабль. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Упражнение 1

Внимание. Внимание в заданиях есть код, который потенциально содержит ошибки, вам нужно объяснить ошибки и представить рабочую версию кода.

Вам нужно написать тестовый класс, чтобы самостоятельно это проверить,необходимо объяснить полученные результаты и связать их с понятием ООП - полиморфизм. Некоторые объявления могут вызвать ошибки компиляции. *Объясните полученные ошибки*, если таковые имеются.

Листинг 5.2 – Пример для выполнения

```
Shape s1 = new Circle(5.5, "RED", false); // Upcast
Circle to ShapeSystem.out.println(s1);//which version?
    System.out.println(s1.getArea());// which version?
System.out.println(s1.getPerimeter());//which version?
System.out.println(s1.getColor());
System.out.println(s1.isFilled());
System.out.println(s1.getRadius());

Circle c1 = (Circle)s1;// Downcast back to Circle
System.out.println(c1);
System.out.println(c1.getArea());
System.out.println(c1.getPerimeter());
System.out.println(c1.getColor());
```

```
System.out.println(c1.isFilled());
System.out.println(c1.getRadius());
Shape s2 = new Shape();
Shape s3 = new Rectangle(1.0, 2.0, "RED", false);//
Upcast System.out.println(s3);
System.out.println(s3.getArea());
System.out.println(s3.getPerimeter());
System.out.println(s3.getColor());
System.out.println(s3.getLength());

Rectangle r1 = (Rectangle)s3; //downcast
System.out.println(r1);
System.out.println(r1.getArea());
System.out.println(r1.getColor());
System.out.println(r1.getLength());
Shape s4 = new Square(6.6);//Upcast
System.out.println(s4);
System.out.println(s4.getArea());
System.out.println(s4.getColor());
System.out.println(s4.getSide());
/*обратите внимание, что выполняем downcast Shape s4 к
Rectangle, который является суперклассом
Square(родителем), вместо Square*//
Rectangle r2 = (Rectangle)s4;
System.out.println(r2);
System.out.println(r2.getArea());
System.out.println(r2.getColor());
System.out.println(r2.getSide());
System.out.println(r2.getLength());
// Downcast Rectangle r2 к Square
Square sq1 = (Square)r2;
System.out.println(sq1);
System.out.println(sq1.getArea());
System.out.println(sq1.getColor());
System.out.println(sq1.getSide());
System.out.println(sq1.getLength());
```