Абстрактные классы

Абстрактные классы

Абстрактный класс в объектно-ориентированном программировании — базовый класс, который не предполагает создания экземпляров. Абстрактные классы представляют собой шаблон для последующей реализации с помощью обычных классов. Абстрактные классы часто используются для планирования как архитектуры приложения в целом, так и для отдельных частей функциональности.

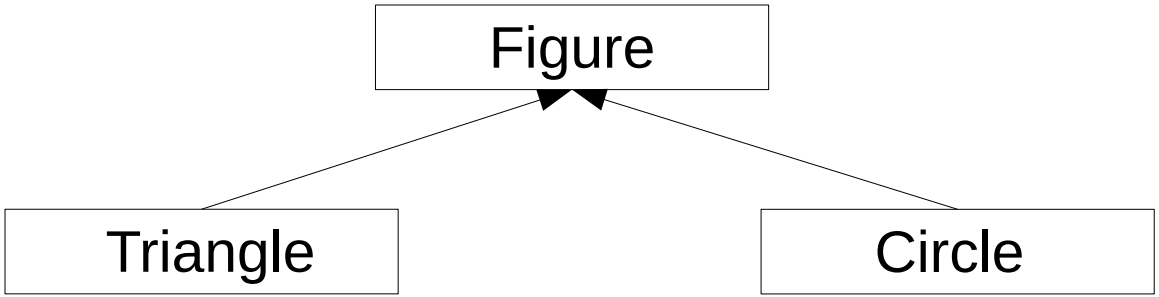
Для чего используются абстрактные классы

Абстрактные классы выступают в роли шаблонов классов которые обязательны к реализации всеми наследующими классами. При описании абстрактных классов в них задаются методы которые должны присутствовать во всех подклассах. Тем самым задается поведение свойственное создаваемой иерархии классов.

Пример использования абстрактных классов

Предположим, что реализуется небольшое приложение использующие различные геометрические фигуры. Требуемым поведением каждой фигуры является вычисление периметра и площади каждой фигуры. Для этого можно использовать абстрактные классы. Т.е. ввести класс Figure в котором и будут объявлены эти методы. После этого создать иерархию классов для описания конкретных фигур.

Пример иерархии классов



Класс Figure будет абстрактным, а классы Triangle, Circle будут подклассами Figure.

Как объявляются абстрактные классы

В Java абстрактные классы объявляются с помощью ключевого слово **abstract**. Абстрактные классы могут содержать один или более абстрактный метод. Абстрактный метод - не статический метод класса в котором описана сигнатура, но отсутствует реализация. Для объявления используется ключевое слово **abstract**. Не запрещено объявлять абстрактные классы без абстрактных методов (но тогда их использование становиться бессмысленным).

Абстрактные классы могут содержать:

- Поля
- Обычные методы
- Конструкторы
- Статические методы

Пример описания абстрактного класса

```
public abstract class Figure {

    public abstract double getArea();

    public abstract double getPerimeter();

    @Override
    public String toString() {
        return "Figure []";
    }

}
```

Абстрактные методы

Обычный метод

При объявлении класса используется ключевое слово abstract. Это же ключевое слово использовалось при объявлении абстрактных методов.

Важные моменты при объявлении абстрактного класса

При объявлении абстрактного класса нужно помнить о следующих особенностях:

1) Статические методы не могут быть абстрактными
2) Методы с модификатором private не могут быть абстрактными
3) Желательно не объявлять абстрактные методы с модификатором по умолчанию

Класс Triangle как подкласс Figure

```java
public class Triangle extends Figure {
	private double sideA;
	private double sideB;
	private double sideC;

	public Triangle(double sideA, double sideB, double sideC) {
		this.sideA = sideA;
		this.sideB = sideB;
		this.sideC = sideC;
	}

	public Triangle() {
		super();
	}

	public double getSideA() {
		return sideA;
	}
	public void setSideA(double sideA) {
		this.sideA = sideA;
	}
	public double getSideB() {
		return sideB;
	}
	public void setSideB(double sideB) {
		this.sideB = sideB;
	}
	public double getSideC() {
		return sideC;
	}
	public void setSideC(double sideC) {
		this.sideC = sideC;
	}

	@Override
	public double getArea() {
		double halfPerimeter = getPerimeter() / 2.0;
		return Math.sqrt(halfPerimeter * (halfPerimeter - sideA) * (halfPerimeter - sideB) * (halfPerimeter - sideC));
	}

	@Override
	public double getPerimeter() {
		return sideA + sideB + sideC;
	}

	@Override
	public String toString() {
		return "Triangle [sideA=" + sideA + ", sideB=" + sideB + ", sideC=" + sideC + "]";
	}
}
```

Реализация абстрактных методов

Важные моменты при наследовании от абстрактного класса

При наследовании от абстрактного класса нужно помнить о следующих особенностях:

1) Ваш класс не будет компилироваться, до тех пор пока вы не реализуете все абстрактные методы или не объявите свой класс абстрактным.
2) Обычные методы абстрактного класса наследуются в обычном порядке и могут быть вызваны.

Вопросы и ответы к ним

Если нельзя создать объект абстрактного класса, то можно ли создать ссылку типа абстрактного класс?

Да. Тут будет работать ровно такой же механизм как и при обычном наследовании.

К каким членам класса можно обратиться используя ссылку типа абстрактного класса?

Как и при обычном наследовании только к тому, что доступно из ее типа.

При переопределении абстрактных методов используется термин реализация методов почему?

Действительно переопределение абстрактных методов называется реализацией Вы реализуете поведение заданное заданное с помощью абстрактного класса.

Пример работы со ссылками типа абстрактного класса

```
public class Main {

    public static void main(String[] args) {

        Figure fig1 = new Triangle(3, 4, 5);

        System.out.println(fig1.getArea());

    }

}
```

Использование ссылки типа абстрактный класс

Вызов методов с помощью этой ссылки

Пример работы со ссылками типа абстрактного класса

```
public class GameBoard {
	private Figure[] figures = new Figure[4];

	public void addFigure(Figure fig) {
		for (int i = 0; i < figures.length; i++) {
			if (figures[i] == null) {
				figures[i] = fig;
				break;
			}
		}
	}

	public double getTotalArea() {
		double area = 0;
		for (Figure figure : figures) {
			if (figure != null) {
				area += figure.getArea();
			}
		}
		return area;
	}

	@Override
	public String toString() {
		String result = "GameBoard [figures=";
		for (Figure figure : figures) {
			if (figure != null) {
				result += ", " + figure.toString();
			}
		}
		result = result.substring(0, result.length() - 2);
		return result + "]";
	}

}
```

Использование ссылки типа абстрактный класс

Использование ссылки типа абстрактный класс

Пример работы со ссылками типа абстрактного класса

```
public class Main {

     public static void main(String[] args) {

          Triangle triangle = new Triangle(3, 4, 5);

          GameBoard gb = new GameBoard();

          gb.addFigure(triangle);

          System.out.println(gb);

     }
}
```

Использование приведения типа ссылки

И хотя метод добавления фигуры на доску использует ссылку типа Figure при вызове можно использовать ссылку тип которой определяется любым подклассом класса Figure. Именно этот механизм позволяет реализовать масштабирование части функционала.

Список литературы

1) Герберт Шилдт Java 8. Полное руководство 9-е издание ISBN 978-5-8459-1918-2
2) https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
3) https://docs.oracle.com/javase/tutorial/java/IandI/override.html