Язык программирования Java

Мультимедийный курс

автор: Васильев А.Н.

www.vasilev.kiev.ua

Киев 2017

Интерфейсы и абстрактные классы

- Абстрактные методы и абстрактные классы
- Знакомство с интерфейсами
- Реализация интерфейсов в классе
- Интерфейсные переменные
- Методы интерфейсов с кодом по умолчанию
- Расширение интерфейсов
- Наследование классов и реализация интерфейсов

Абстрактные методы и классы

- При описании метода в классе указывается тип результат, название метода и список аргументов, а также тело метода с программным кодом. Но можно ограничиться объявлением метода - указать его сигнатуру, но не описывать тело метода. Соответствующий метод называется абстрактным
- Каждый абстрактный метод описывается с ключевым словом `abstract`
- Класс, в котором есть хотя бы один абстрактный метод, также называется абстрактным
- Абстрактный класс описывается с ключевым словом `abstract`
- Если класс просто описать с ключевым словом `abstract`, то он будет абстрактным, даже если в нем нет абстрактных методов
- На основе абстрактного класса нельзя создать объект. Обычно их используют при наследовании.
- Можно объявить объектную переменную абстрактного класса. Такая переменная может ссылаться на объект любого подкласса, созданного на основе данного абстрактного класса
- Если в подклассе, который создается на основе абстрактного суперкласса, некоторые абстрактные методы оставить не описанными, то такой подкласс также будет абстрактным
- Если класс просто описать с ключевым словом `abstract`, то он будет абстрактным, даже если в нем нет абстрактных методов

Программа: абстрактные классы

Мы создаем несколько классов, которые в определенном смысле описывают разные геометрические фигуры (равносторонний треугольник, квадрат и круг). Мы предполагаем наличие у геометрических фигур таких характеристик, как цвет ("раскрашенная" фигура), линейный размер (для треугольника и квадрата длина стороны, для круга радиус) и площадь

Детали

Если равносторонний треугольник имеет стороны длины R, то площадь треугольника дается выражением $\frac{\sqrt{3}}{4}R^2$. Для квадрата со стороной R площадь вычисляется как R^2. Площадь круга радиуса R вычисляется выражением πR^2, где иррациональная постоянная $\pi \approx 3.141592$. Все три случая можно свести к формуле kR^2, где параметр $k = \frac{\sqrt{3}}{4}$ для треугольника, $k = 1$ для квадрата и $k = \pi$ для круга.

В программе описывается абстрактный класс с названием `ColoredFigure`, на основе которого путем наследования создаются классы `Triangle` (класс для описания треугольника), `Square` (класс для описания квадрата) и `Circle` (класс для описания круга). В главном методе программы создаются объекты производных классов и показано, как получить доступ к объектам через объектные переменные подклассов и через объектную переменную абстрактного суперкласса

Абстрактные классы

Использование абстрактного класса - 1

```
abstract class ColoredFigure{          // Абстрактный суперкласс
   String color;                       // Текстовое поле (цвет)
   int size;                           // Целочисленное поле (размер)
   ColoredFigure(String clr,int s){    // Конструктор
      // Присваивание значений полям:
      color=clr;
      size=s;
   } // Метод для отображения информации об объекте:
   void show(){
      // Цвет и название фигуры:
      System.out.println("Фигура: "+color+" "+getName());
      // Характерный размер:
      System.out.println("Характерный размер ("+getSizeName()+"): "+size);
      // Площадь:
      System.out.printf("Площадь: %.3f\n",getArea());
      // Импровизированная "линия" из "звездочек":
      String line="";
      for(int k=1;k<=30;k++){
         line+="*";
      }
      System.out.println(line);
   } // Метод результатом возвращает название для
     // параметра, определяющего характерный размер фигуры:
   String getSizeName(){
      return "сторона";
   }
   // Абстрактные методы:
   abstract String getName(); // Название фигуры
   abstract double getArea(); // Площадь фигуры
} // Продолжение на следующем слайде!!!
```

Абстрактные классы

Использование абстрактного класса - 2

```
class Triangle extends ColoredFigure{ // Подкласс (фигура - треугольник)
   Triangle(String clr,int s){        // Конструктор
      super(clr,s);                   // Вызов конструктора суперкласса
   }
   // Описание абстрактного метода из суперкласса, возвращающего название фигуры:
   String getName(){
      return "треугольник";
   }
   // Описание абстрактного метода из суперкласса, возвращающего площадь фигуры:
   double getArea(){
      double k=Math.sqrt(3)/4;
      return size*size*k;
   }
}
class Square extends ColoredFigure{   // Подкласс (фигура - квадрат)
   Square(String clr,int s){          // Конструктор
      super(clr,s);                   // Вызов конструктора суперкласса
   }
   // Описание абстрактного метода из суперкласса,
   // возвращающего результатом название фигуры:
   String getName(){
      return "квадрат";
   }
   // Описание абстрактного метода из суперкласса,
   // возвращающего результатом площадь фигуры:
   double getArea(){
      double k=1;
      return size*size*k;
   }
}  // Продолжение на следующем слайде!!!
```

Абстрактные классы

Использование абстрактного класса - 3

```
// Подкласс (фигура - круг):
class Circle extends ColoredFigure{
   // Конструктор:
   Circle(String clr,int s){
   // Вызов конструктора суперкласса:
      super(clr,s);
   }
   // Описание абстрактного метода из суперкласса,
   // возвращающего результатом название фигуры:
   String getName(){
      return "круг";
   }
   // Описание абстрактного метода из суперкласса,
   // возвращающего результатом площадь фигуры:
   double getArea(){
      double k=Math.PI;
      return size*size*k;
   }
   // Переопределение метода, возвращающего результатом
   // название для параметра, определяющего характерный
   // размер фигуры:
   String getSizeName(){
      return "радиус";
   }
}
// Продолжение на следующем слайде!!!
```

Абстрактные классы

Использование абстрактного класса - 4

```
// Главный класс:
class UsingAbstractClassDemo{
   // Главный метод:
   public static void main(String[] args){
      // Создание объектов:
      Triangle T=new Triangle("красный",2); // Треугольник
      Square S=new Square("черный",3);      // Квадрат
      Circle C=new Circle("желтый",1);      // Круг
      // Отображение информации об объектах через
      // объектные переменные подклассов:
      System.out.println("Использование объектных переменных подкласса");
      T.show();
      S.show();
      C.show();
      // Объектная переменная абстрактного суперкласса:
      ColoredFigure F;
      // Отображение информации об объектах через
      // объектную переменную абстрактного суперкласса:
      System.out.println("Использование объектной переменной абстрактного суперкласса");
      F=T;       // Треугольник
      F.show();
      F=S;       // Квадрат
      F.show();
      F=C;       // Круг
      F.show();
   }
}
```

Абстрактные классы

Использование абстрактного класса - результат

```
Использование объектных переменных подкласса
Фигура: красный треугольник
Характерный размер (сторона): 2
Площадь: 1,732
*****************************
Фигура: черный квадрат
Характерный размер (сторона): 3
Площадь: 9,000
*****************************
Фигура: желтый круг
Характерный размер (радиус): 1
Площадь: 3,142
*****************************
Использование объектной переменной абстрактного суперкласса
Фигура: красный треугольник
Характерный размер (сторона): 2
Площадь: 1,732
*****************************
Фигура: черный квадрат
Характерный размер (сторона): 3
Площадь: 9,000
*****************************
Фигура: желтый круг
Характерный размер (радиус): 1
Площадь: 3,142
*****************************
```

Интерфейсы

- Интерфейс напоминает абстрактный класс: интерфейс содержит объявление методов и/или статических констант

- В Java 8 интерфейс может содержать не только объявление методов, но еще и описание методов так называемые версии методов по умолчанию

- Интерфейсы используются путем реализации их в классах

- Класс, который реализует интерфейс, должен содержать описание всех методов, объявленных в интерфейсе

- Один и тот же класс может реализовать одновременно несколько интерфейсов

Реализация интерфейса

- Интерфейс описывается практически так же, как и класс. Только вместо ключевого слова `class` используется ключевое слово `interface`

- Кроме объявления методов, в интерфейсе можно объявлять статические константные поля. Такие поля объявляются как обычные поля со значениями, но автоматически интерпретируются, как если бы они были описаны с ключевыми словами `static` и `final`

- Методы, объявленные в интерфейсе, являются `public`-методами (хотя при объявлении методов в интерфейсе ключевое слово `public` не используется)

Пример описания интерфейса

```
interface MyInterface{
  int NUMBER=100;
  int getNumber(double x);
  char getSymbol(int n);
}
```

Пример описания класса

```
class MyClass implements MyInterface{
  public int getNumber(double x){
    return (int)x;
  }
  public char getSymbol(int n){
    return (char)('A'+n);
  }
}
```

Реализация интерфейса

Пример реализации интерфейса

```
// Интерфейс:
interface MyInterface{
   // Статическая константа:
   int NUMBER=100;
   // Объявление методов:
   int getNumber(double x);
   char getSymbol(int n);
}
// Класс реализует интерфейс:
class MyClass implements MyInterface{
   // Описание методов:
   public int getNumber(double x){
      return (int)x;
   }
   public char getSymbol(int n){
      return (char)('A'+n);
   }
} // Главный класс:
class UsingInterfaceDemo{
   public static void main(String[] args){
      MyClass obj=new MyClass();
      System.out.println("Статическая константа: "+MyClass.NUMBER);
      System.out.println("Целое число: "+obj.getNumber(12.5));
      System.out.println("Символ: "+obj.getSymbol(3));
   }
}
```

Результат

```
Статическая константа: 100
Целое число: 12
Символ: D
```

Реализация интерфейса

Пример реализации нескольких интерфейсов

```
// Первый интерфейс:
interface First{
   void hello();
} // Второй интерфейс:
interface Second{
   void hi();
} // Класс реализует два интерфейса:
class MyClass implements First, Second{
   // Описание метода из первого интерфейса:
   public void hello(){
      System.out.println("Метод из интерфейса First");
   }
   // Описание метода из второго интерфейса:
   public void hi(){
      System.out.println("Метод из интерфейса Second");
   }
} // Главный класс:
class UsingInterfacesDemo{
   public static void main(String[] args){
      // Создание объекта:
      MyClass obj=new MyClass();
      // Вызов методов из объекта:
      obj.hello();
      obj.hi();
   }
}
```

Результат

```
Метод из интерфейса First
Метод из интерфейса Second
```

Интерфейсные переменные

- **Интерфейсная переменная - переменная, "тип" которой определяется названием интерфейса**
- **Интерфейсная переменная может ссылаться на объект класса, реализующего данный интерфейс**

Пример реализации нескольких интерфейсов - 1

```
// Интерфейс:
interface Base{
    // Объявление метода:
    void show();
}
// Класс реализует интерфейс Base:
class Alpha implements Base{
    // Текстовое поле:
    String word;
    // Конструктор:
    Alpha(String txt){
        word=txt;
    }
    // Описание метода из интерфейса:
    public void show(){
        System.out.println("Объект класса Alpha");
        System.out.println("Текстовое поле: "+word);
    }
} // Продолжение на следующем слайде!!!
```

Интерфейсные переменные

Пример реализации нескольких интерфейсов - 2

```
// Класс реализует интерфейс Base:
class Bravo implements Base{
   // Целочисленное поле:
   int number;
   // Конструктор:
   Bravo(int n){
      number=n;
   }
   // Описание метода из интерфейса:
   public void show(){
      System.out.println("Объект класса Bravo");
      System.out.println("Целочисленное поле: "+number);
   }
} // Главный класс:
class UsingInterfaceVarsDemo{
   public static void main(String[] args){
      // Интерфейсная переменная:
      Base ref;
      // Объект класса Alpha:
      ref=new Alpha("текст");
      ref.show();
      // Объект класса Bravo:
      ref=new Bravo(123);
      ref.show();
   }
}
```

Результат

```
Объект класса Alpha
Текстовое поле: текст
Объект класса Bravo
Целочисленное поле: 123
```

Методы с кодом по умолчанию

- Метод в интерфейсе можно не только объявить, но и описать
- Если метод описан в интерфейсе, то в классе, реализующем соответствующий интерфейс, метод можно не описывать
- Методы с кодом по умолчанию описываются в интерфейсе с ключевым словом `default`

Метод интерфейса с кодом по умолчанию - 1

```
// Интерфейс с методом, имеющим код по умолчанию:
interface Base{
    // Метод с кодом по умолчанию:
    default void show(String txt){
        System.out.println("Интерфейс Base: "+txt);
    }
    // Объявление метода:
    void hello();
} // Класс реализует интерфейс Base:
class Alpha implements Base{
    // Описание обычного метода:
    public void hello(){
        System.out.println("Объект класса Alpha");
    }
    // Описание метода с кодом по умолчанию:
    public void show(String txt){
        System.out.println("Класс Alpha: "+txt);
    }
} // Продолжение на следующем слайде!!!
```

Методы с кодом по умолчанию

Метод интерфейса с кодом по умолчанию - 2

```
class Bravo implements Base{    // Класс реализует интерфейс Base
   public void hello(){         // Описание обычного метода
      System.out.println("Объект класса Bravo");
   }
} // Главный класс:
class UsingDefaultMethodsDemo{
   public static void main(String[] args){
      Base ref;                 // Интерфейсная переменная
      Alpha objA=new Alpha();   // Объект класса Alpha
      // Вызов методов через объектную переменную:
      objA.hello();
      objA.show("объектная переменная objA");
      // Интерфейсной переменной присваивается ссылка на объект класса Alpha:
      ref=objA;
      // Вызов метода через интерфейсную переменную:
      ref.show("интерфейсная переменная ref");
      Bravo objB=new Bravo();   // Объект класса Bravo
      // Вызов методов через объектную переменную:
      objB.hello();
      objB.show("объектная переменная objB");
      // Интерфейсной переменной присваивается ссылка на объект класса Bravo:
      ref=objB;
      // Вызов метода через интерфейсную переменную:
      ref.show("интерфейсная переменная ref");
   }
}
```

Методы с кодом по умолчанию

Метод интерфейса с кодом по умолчанию - результат

```
Объект класса Alpha
Класс Alpha: объектная переменная objA
Класс Alpha: интерфейсная переменная ref
Объект класса Bravo
Интерфейс Base: объектная переменная objB
Интерфейс Base: интерфейсная переменная ref
```

Методы с кодом по умолчанию

Детали

В некоторых случаях приходится в явном виде вызывать версию метода, описанную в интерфейсе. Это можно сделать. Ссылка на версию метода из интерфейса выполняется в таком формате: указывается имя интерфейса, точка, ключевое слово `super`, точка и, собственно, инструкция вызова метода. Например, если нужно вызвать версию метода `hello()` из интерфейса `First`, то соответствующая инструкция выглядит как `First.super.hello()`. Аналогично, инструкция `Second.super.hello()` означает вызов версии метода `hello()` из интерфейса `Second`.

Методы с кодом по умолчанию

Вызов версии метода из интерфейса

```
// Первый интерфейс:
interface First{
   default void hello(){
      System.out.println("Метод из интерфейса First");
   }
} // Второй интерфейс:
interface Second{
   default void hello(){
      System.out.println("Метод из интерфейса Second");
   }
} // Класс реализует два интерфейса:
class MyClass implements First, Second{
   public void hello(){            // Описание метода
      // Вызов версии метода из интерфейса First:
      First.super.hello();
      // Вызов версии метода из интерфейса Second:
      Second.super.hello();
   }
} // Главный класс:
class MoreDefaultMethodsDemo{
   public static void main(String[] args){
      MyClass obj=new MyClass();  // Создание объекта
      // Вызов метода:
      obj.hello();
   }
}
```

Результат

```
Метод из интерфейса First
Метод из интерфейса Second
```

Расширение интерфейсов

- Наследование применимо не только к классам, но и к интерфейсам: один интерфейс может наследовать другой интерфейс
- Наследование интерфейсов (называется расширением интерфейса) реализуется так же, как и наследование классов: в описании интерфейса-наследника после его имени указывается ключевое слово `extends`, после которого указывается имя наследуемого интерфейса

Расширение интерфейсов - 1

```
// Наследуемый интерфейс:
interface First{
   // Метод с кодом по умолчанию:
   default void alpha(){
      System.out.println("Интерфейс First: метод alpha()");
   }
   // Метод с кодом по умолчанию:
   default void bravo(){
      System.out.println("Интерфейс First: метод bravo()");
   }
   // Метод с кодом по умолчанию:
   default void charlie(){
      System.out.println("Интерфейс First: метод charlie()");
   }
   // Метод без кода по умолчанию:
   void delta();
}
// Продолжение на следующем слайде!!!
```

Расширение интерфейсов

Расширение интерфейсов - 2

```
// Интерфейс-наследник:
interface Second extends First{
   // Объявление метода:
   void bravo();
   // Метод с кодом по умолчанию:
   default void charlie(){
      System.out.println("Интерфейс Second: метод charlie()");
   }
   // Объявление метода:
   void echo();
}
// Класс реализует интерфейс:
class MyClass implements Second{
   // Описание методов:
   public void bravo(){
      System.out.println("Класс MyClass: метод bravo()");
   }
   public void delta(){
      System.out.println("Класс MyClass: метод delta()");
   }
   public void echo(){
      System.out.println("Класс MyClass: метод echo()");
   }
}
// Продолжение на следующем слайде!!!
```

Расширение интерфейсов

Расширение интерфейсов - 3

```
// Главный класс:
class ExtendingInterfaceDemo{
   public static void main(String[] args){
      // Создание объекта:
      MyClass obj=new MyClass();
      // Вызов методов:
      obj.alpha();
      obj.bravo();
      obj.charlie();
      obj.delta();
      obj.echo();
   }
}
```

Результат

```
Интерфейс First: метод alpha()
Класс MyClass: метод bravo()
Интерфейс Second: метод charlie()
Класс MyClass: метод delta()
Класс MyClass: метод echo()
```

Расширение интерфейсов

Расширение интерфейсов

```
interface First{              // Первый интерфейс
   default void alpha(){      // Описание метода
      System.out.println("Интерфейс First: метод alpha()");
   }
} // Второй интерфейс:
interface Second extends First{
   default void alpha(){      // Описание методов
      First.super.alpha(); // Вызов версии метода из интерфейса First
      System.out.println("Интерфейс Bravo: метод alpha()");
   }
   default void bravo(){
      System.out.println("Интерфейс Bravo: метод bravo()");
   }
} // Класс реализует интерфейс:
class MyClass implements Second{
   public void bravo(){       // Описание метода
      Second.super.bravo(); // Вызов версии метода из интерфейса Second
      System.out.println("Класс MyClass: метод bravo()");
   }
} // Главный класс:
class MoreExtendingInterfaceDemo{
   public static void main(String[] args){
      MyClass obj=new MyClass(); // Создание объекта
      obj.alpha(); // Вызов методов
      obj.bravo();
   }
}
```

```
Интерфейс First: метод alpha()
Интерфейс Bravo: метод alpha()
Интерфейс Bravo: метод bravo()
Класс MyClass: метод bravo()
```

Наследование класса и реализация интерфейсов

Класс может наследовать суперкласс и одновременно реализовывать несколько интерфейсов

Наследование класса и реализация интерфейсов - 1

```
// Первый интерфейс:
interface First{
   String getFirst();         // Объявление метода
   default void show(){       // Описание метода
      System.out.println("Интерфейс First: метод show()");
   }
} // Второй интерфейс:
interface Second{
   String getSecond();      // Объявление метода
   default void show(){     // Описание метода
      System.out.println("Интерфейс Second: метод show()");
   }
} // Суперкласс:
class Base{
   // Описание метода:
   String getBase(){
      return "Класс Base";
   }
   // Описание метода:
   void show(){
      System.out.println("Класс Base: метод show()");
   }
} // Продолжение на следующем слайде!!!
```

Наследование класса и реализация интерфейсов - 2

```
// Подкласс наследует суперкласс и реализует интерфейсы:
class MyClass extends Base implements First, Second{
   public String getFirst(){    // Описание метода
      return "Интерфейс First";
   } // Описание метода:
   public String getSecond(){
      return "Интерфейс Bravo";
   } // Описание метода:
   public void show(){
      System.out.println("Мы используем:");
      System.out.println(getFirst());
      System.out.println(getSecond());
      System.out.println(getBase());
      First.super.show();       // Вызов версии метода из интерфейса First
      Second.super.show();      // Вызов версии метода из интерфейса Second
      super.show();             // Вызов версии метода из суперкласса Base
   }
} // Главный класс:
class ExtendingAndImplementingDemo{
   public static void main(String[] args){
      // Создание объекта подкласса:
      MyClass obj=new MyClass();
      // Вызов метода из объекта:
      obj.show();
   }
}
```

Результат

```
Мы используем:
Интерфейс First
Интерфейс Bravo
Класс Base
Интерфейс First: метод show()
Интерфейс Second: метод show()
Класс Base: метод show()
```

Домашнее задание

- Проанализировать программный код примеров, представленных в презентации
- Набрать код и проверить его функциональность

Продолжение следует...