Java – Переопределение (overriding)

В предыдущей главе мы рассказали про суперклассы и подклассы. Если класс наследует метод из своего суперкласса, тогда есть шанс переопределить взятый метод, если он не помечен final.

Так что такое в Java overriding или override – это **переопределение.**

Преимущество в Java переопределения заключается в том, что оно позволяет определять (описывать) поведение, характерное для типа подкласса, значит подкласс может реализовать метод родительского класса на основе его требования.

В объектно-ориентированных терминах, переопределение значит перезапись функционала существующего метода.

Пример 1

Рассмотрим пример.

```
class Animal {
   public void move() {
      System.out.println("Животные могут двигаться");
   }
}

class Dog extends Animal {
   public void move() {
      System.out.println("Собаки могут ходить и бегать");
   }
}

public class TestDog {

   public static void main(String args[]) {
      Animal a = new Animal();   // Animal - ссылка и объект
      Animal b = new Dog();   // Animal - ссылка, но Dog - объект

      a.move();   // Запускает метод в классе Animal
      b.move();   // Запускает метод в классе Dog
   }
}
```

После запуска программы будет выдан такой результат:

```
Животные могут двигаться
Собаки могут ходить и бегать
```

В вышеприведённом примере вы можете заметить, что b хоть и является типом Animal, оно запускает метод move в классе Dog. Причина тому: во время компиляции проходит проверка ссылочного типа. Однако, во время выполнения, JVM определяет тип объекта и запускает метод, который принадлежит этому конкретному объекту.

Следовательно, по примеру выше, программа запустится правильно, так как класс Animal имеет метод move. Затем, во время выполнения, он запускает метод, принадлежащий этому объекту.

Рассмотрите следующий пример:

Пример 2

```
class Animal {
   public void move() {
      System.out.println("Животные могут двигаться");
   }
}
```

```
class Dog extends Animal {
   public void move() {
      System.out.println("Собаки могут ходить и бегать");
   }
   public void bark() {
      System.out.println("Собаки могут лаять");
   }
}

public class TestDog {

   public static void main(String args[]) {
      Animal a = new Animal();   // Animal - ссылка и объект
      Animal b = new Dog();   // Animal - ссылка, но Dog - объект

      a.move();   // Запускает метод в классе Animal
      b.move();   // Запускает метод в классе Dog
      b.bark();
   }
}
```

После запуска программы будет выдан такой результат:

```
TestDog.java:26: error: cannot find symbol
      b.bark();
       ^
  symbol:   method bark()
  location: variable b of type Animal
1 error
```

Программа выдаст ошибку во время компиляции, так как ссылочный тип b у Animal не имеет метода под именем bark.

Правила переопределения метода

- Список аргументов должен быть точно таким же, как и для переопределённого метода.
- Возвращаемый тип должен быть таким же или подтипом возвращаемого типа, объявленного в исходном переопределенном методе в суперклассе.
- Уровень доступа не может быть более ограниченным, чем уровень доступа переопределённого метода. Например, если метод суперкласса объявлен public, то переопределяемый метод в подклассе не может быть private или protected.
- Методы экземпляров могут быть переопределены только если они наследованы подклассом.
- Методы, которые объявлены как final, не могут быть переопределены.
- Статические методы, которые объявлены как static, не могут быть переопределены, но могут быть повторно объявлены.
- Если метод нельзя наследовать, то его нельзя переопределить.
- Подкласс внутри того же пакета, что и суперкласс экземпляра, может переопределять любой метод суперкласса, который не объявлен как private или final.
- Подкласс в другом пакете может переопределять только не final методы, объявленные как public или protected.
- Переопределяемый метод может выдавать любые непроверенные исключения вне зависимости от того, переопределяет ли переопределённый метод какие-либо непроверенные исключения или нет. Однако, переопределяемый метод не должен генерировать проверенные исключения, которые являются новыми или более широкими, чем те, которые объявлены переопределённым методом. Переопределенный метод может генерировать более узкие или меньшие исключения, чем переопределенный метод.
- Конструкторы нельзя переопределить.

Использование ключевого слова super

Вызывая версию суперкласса переопределённого метода, используется ключевое слово super.

Пример

```
class Animal {
   public void move() {
      System.out.println("Животные могут двигаться");
   }
}
```

```
class Dog extends Animal {
   public void move() {
      super.move();   // Вызывает метод суперкласса
      System.out.println("Собаки могут ходить и бегать");
   }
}

public class TestDog {

   public static void main(String args[]) {
      Animal b = new Dog();   // Animal - ссылка, но Dog - объект
      b.move();   // Запуск метода в классе Dog
   }
}
```

После запуска программы будет выдан такой результат:

```
Животные могут двигаться
Собаки могут ходить и бегать
```