Кравченко Даниил Андреевич
бакалавр, студент
ФГБОУ ВО «Государственный университет морского
и речного флота им. адмирала С.О. Макарова»
г. Санкт-Петербург

REFLECTION В JAVA

Аннотация*: статья посвящена понятию reflection, которое на русский язык можно перевести, как отражение и связанными с ней процессами в объектно-ориентированном языке программирования Java. Приведены и описаны такие понятия как, reflection и ее возможности. Приведены требуемые для ее работы класс и дополнительные методы. Для наглядной демонстрации возможностей написан пример простой программы и записаны важные по ней наблюдения.*

Ключевые слова*: reflection, поведение, во время выполнения, объект, метод, интерфейс, класс, спецификатор доступа.*

Reflection – это Java API.

Reflection API состоит из двух компонентов: объектов, которые представляют различные части файла класса, и средства для безопасного и надежного извлечения этих объектов. Последнее очень важно, поскольку Java предоставляет множество мер безопасности, и было бы нецелесообразно предоставлять набор классов, которые делают эти меры недействительными.

Первый компонент – Reflection API – это механизм, используемый для получения информации о классе. Этот механизм встроен в класс с именем Class. Специальный класс Class – это универсальный тип метаинформации, описывающей объекты в системе Java. Загрузчики классов в системе Java возвращают объекты типа Class. До сих пор тремя наиболее интересными методами в этом классе были:

– forName, который загрузит класс с заданным именем, используя текущий загрузчик классов;

– getName, который возвращал бы имя класса как String объект, что было полезно для идентификации ссылок на объекты по их имени класса;

– newInstance, который вызовет нулевой конструктор в классе (если он существует) и вернет вам экземпляр объекта этого класса объекта.

К этим трем полезным методам Reflection API добавляет в класс несколько дополнительных методов Class. Это следующие:

– getConstructor, getConstructors,getDeclaredConstructor;

– getMethod, getMethods,getDeclaredMethods;

– getField, getFields,getDeclaredFields;

– getSuperclass;

– getInterfaces;

– getDeclaredClasses.

В дополнение к этим методам было добавлено много новых классов для представления объектов, которые будут возвращать эти методы. Новые классы в основном являются частью пакета java.lang.reflect, но некоторые из новых базовых классов типа (Void, Byte и т. д.) находятся в пакете java.lang. Было принято решение разместить новые классы там, где они есть, путем помещения классов, представляющих метаданные, в пакет reflection и классов, представляющих типы в языковом пакете.

Таким образом, Reflection API представляет ряд изменений в классе, Class которые позволяют задавать вопросы о внутреннем устройстве класса, и набор классов, которые представляют ответы, которые дают вам эти новые методы.

Пример простой Java программы для демонстрации использования Reflection представлен на рисунке 1.

```java
import java.lang.reflect.Method;
import java.lang.reflect.Field;
import java.lang.reflect.Constructor;
// класс, объект которого должен быть создан
class Test{
    // создание частного поля
    private String s;
    // создание общедоступного конструктора
    public Test()  {  s = "something"; }
    // Создание общедоступного метода без аргументов
    public void method1()  {
        System.out.println("The string is " + s);
    }
    // Создание общедоступного метода с аргументом int
    public void method2(int n)  {
        System.out.println("The number is " + n);
    }
    // создание частного метода
    private void method3() {
        System.out.println("Private method invoked");
    }
}
class Demo {
    public static void main(String args[]) throws Exception {
        // Создание объекта, свойство которого нужно проверить
        Test obj = new Test();
        // Создание объекта класса из объекта с помощью метода getclass
        Class cls = obj.getClass();
        System.out.println("The name of class is " +
                           cls.getName());
        // Получение конструктора класса через объект класса
        Constructor constructor = cls.getConstructor();
        System.out.println("The name of constructor is " +
                           constructor.getName());
        System.out.println("The public methods of class are : ");
        // Получение методов класса через объект класса с помощью getMethods
        Method[] methods = cls.getMethods();
        // Печать имен методов
        for (Method method:methods)
            System.out.println(method.getName());
        // создает объект желаемого метода, предоставляя
        // имя метода и класс параметра как аргументы для
        // getDeclaredMethod
        Method methodcall1 = cls.getDeclaredMethod("method2", int.class);
        // вызывает метод во время выполнения
        methodcall1.invoke(obj, 19);
        // создает объект желаемого поля, предоставляя
        // имя поля в качестве аргумента
        // метод getDeclaredField
        Field field = cls.getDeclaredField("s");
        // разрешает объекту доступ к полю независимо от
        // спецификатора доступа, используемого с полем
        field.setAccessible(true);
        // принимает объект и новое значение для присвоения
        // в поле как аргументы
        field.set(obj, "JAVA");
        // Создает объект желаемого метода, предоставляя
        // имя метода в качестве аргумента для getDeclaredMethod
        Method methodcall2 = cls.getDeclaredMethod("method1");
        // вызывает метод во время выполнения
        methodcall2.invoke(obj);
        // Создает объект желаемого метода, предоставляя
        // имя метода в качестве аргумента
        // метод getDeclaredMethodd
        Method methodcall3 = cls.getDeclaredMethod("method3");
        // разрешает объекту доступ к методу независимо от
        // спецификатора доступа, используемого с методом
        methodcall3.setAccessible(true);
        // вызывает метод во время выполнения
        methodcall3.invoke(obj);
    }
}
```

Рис. 1. Пример Java программы

Важные наблюдения:

1. Мы можем вызвать метод через reflection, если знаем его имя и типы параметров. Ниже мы используем два метода для этой цели. getDeclaredMethod (): для создания объекта вызываемого метода. invoke (): чтобы вызвать метод класса во время выполнения;

2. Через reflection мы можем получить доступ к закрытым переменным и методам класса с помощью его объекта класса и вызвать метод, используя объект, как описано выше. Для этого мы используем ниже два метода. Class.getDeclaredField (FieldName): используется для получения частного поля. Возвращает объект типа Поле для указанного имени поля. Field.setAccessible (true): разрешает доступ к полю независимо от модификатора доступа, используемого с полем.

Список литературы

1. Форман И.Р. Java Reflection In Action / Форман И.Р. – Канада, 2005. – С. 4–198.

2. Документация Java Standard Edition [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/8/docs/

3. Использование Java Reflection [Электронный ресурс]. – Режим доступа: https://www.oracle.com/technical-resources/articles/java/javareflection.html

4. Документация The Reflection API [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/tutorial/reflect/index.html