ТИПИ ВИЩОГО ПОРЯДКУ У МОВІ ПРОГРАМУВАННЯ JAVA

Р.К. ЗАКОЛЕНКО

Мова програмування Java є однією з найпоширеніших мов у світі розробки програмного забезпечення, завдяки своїй надійності, незалежності від платформи та комплексній екосистемі. Узагальнення в Java надають можливість класу, інтерфейсу або методу працювати з довільним типом, зберігаючи відповідний механізм безпеки на етапі компіляції. Введені у Java 5 [1], узагальнення були спрямовані на усунення помилок типів під час компіляції, тим самим підвищуючи безпеку, можливість повторного використання та читабельність коду.

Конструктор типів - це властивість типізованої формальної мови, яка слугує для створення нових типів на базі інших. Наприклад, у Java, `List` - це конструктор типів. Коли йому дано параметр типу, скажімо, Int, він створює новий тип `List[Int]`. У свою чергу, типи вищого порядку дозволяють конструкторам типів приймати інші конструктори типів як параметри. Ця концепція добре відома в мовах, таких як Scala, де вона використовується для створення узагальненого та повторно використовуваного коду:

```
trait Functor[F[_]]:
  def map[A, B](fa: F[A])(f: A => B): F[B]

object OptionFunctor extends Functor[Option]:
  def map[A, B](fa: Option[A])(f: A => B): Option[B] =
    fa match
      case Some(a) => f(a)
      case None => None
```

У прикладі вище на Scala, `Functor` - це `trait` (схожий на інтерфейс у Java) з параметром, який є конструктором типу - `F[_]`. Функція `map`, визначена в `Functor`, є загальною для двох типів `A` та `B` і абстрагує будь-який конструктор типу `F`. Об'єкт `OptionFunctor` надає конкретну реалізацію `Functor` для `Option`. На противагу цьому, система типів та синтаксис Java не підтримують цей рівень абстракції з конструкторами типів. Для забезпечення інтеграції типів вищого роду, потрібно внести зміни до синтаксису мови та її системи типів.

Хоча Java не підтримує типів вищого роду за замовчуванням, є дослідження, які імітують цю функцію. Бібліотека *Hkt* реалізує механізм легкого поліморфізму вищого порядку [2] за допомогою процесора анотацій. Цей підхід не є дуже ефективним, адже іноді відбувається надмірне створення об'єктів. До того ж він вимагає написання надлишкового синтетичного коду, який може бути важким для розуміння та підтримки. У *Adding Type Constructor Parameterization to Java* презентується числення під назвою FGJ [3], яке формалізує систему типів, яка дозволяє параметризувати класи та методи конструкторами типів. Проте наразі немає відповідної реалізації компілятора Java.

⟨Параметри типу⟩ ::= "<" ⟨Параметр типу⟩ { "," ⟨Параметр типу⟩ } ">"

⟨Параметр типу⟩ ::= ⟨Ідентифікатор⟩ [⟨Параметри типу⟩]

Наведена вище EBNF-нотація *(розширена нотація Бекуса–Наура)* визначає синтаксис, за допомогою якого можна описувати конструкції типів вищого порядку у мові Java, що є кроком до розширення мови та інтеграції типів вищого порядку в її систему типів. Відповідно до цієї нотації визначення `Functor` у Java може мати наступний вигляд:

```
public interface Functor<F<_>> {
   <A, B> F<B> map(F<A> fa, Function<A, B> f);
}
```

Література

[1] G. Bracha, M. Odersky, D. Stoutamire, P. Wadler. *Making the Future Safe for the Past: Adding Genericity to the Java TM Programming Language.* In Functional and Logic Programming, edited by Michael Codish and Eijiro Sumii, 8475:119–35. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2004.

[2] J. Yallop and L. White. *Lightweight Higher-Kinded Polymorphism.* University of Cambridge.

[3] P. Altherr and V. Cremet. *Adding type constructor parameterization to Java.* Accepted to the workshop on Formal Techniques for Java-like Programs (FTfJP'07) at the European Conference on Object-Oriented Programming (ECOOP), 2007.

[4] J. Gosling, B. Joy, G. Steele, and G. Bracha. *Java(TM) Language Specification, The (3rd Edition).* Addison-Wesley Professional, 2005.

[5] D. Gregor, J. Jarvi, J. G. Siek, B. Stroustrup, G. D. Reis, and A. Lumsdaine. Concepts: linguistic support for generic programming in C++. In P. L. Tarr and W. R. Cook (Eds.), OOPSLA, pages 291–310. ACM, 2006.

Національний університет "Києво-Могилянська академія" , Київ, Україна
Email address: r.zakolenko@ukma.edu.ua