

Contents

	Page
Editorial	3
Object-Oriented Programming Languages and Systems (OOPS) Special Track at the 21st ACM Symposium on Applied Computing (SAC 2006)	

ARTICLES

Interaction between Objects in powerJava

By Matteo Baldoni, Guido Boella and Leendert van der Torre

5

This paper shows how the notion of role, as given in ontologies and knowledge representation, can be introduced in Java, defining a prototype language powerJava where roles allow objects to offer different operations depending on the type of the role, of the type and identity of the player of the role, and to define session-aware interaction.

Method overloading and overriding cause distribution transparency and encapsulation flaws

By Antoine Beugnard, and Salah Sadou

31

Based on an experiment using three languages under .NET, this paper shows that overriding and overloading are not compatible with encapsulation; a solution is proposed to avoid exposure of the internal structure of components.

Union Types for Object-Oriented Programming

By Atsushi Igarashi and Hideshi Nagira

47

The authors propose union types for statically typed class-based object-oriented languages as a means to enhance the flexibility of subtyping, allowing the programmer to implement heterogeneous collections and to group independently developed classes with similar interfaces. The notion is formalized on top of Featherweight and a proof of soundness is provided.

Just: safe unknown types in Java-like languages*By Giovanni Lagorio and Elena Zucca*

68

The authors propose an extension of Java-like languages allowing programmers to omit type annotations in strategic places of their programs without losing type-safety. This is achieved by means of inferred type constraints describing the implicit requirements on untyped code to be correctly executed. A prototype implementation is available on a small, yet significant, Java subset.

The Infer Type Refactoring and its Use for Interface-Based Programming*By Friedrich Steimann*

99

The author presents a new refactoring named "Infer Type", which by using type inference completely automates the construction of new, context-specific interfaces and their use in variable declarations, thus supporting greater decoupling and access protection of code, and serving the goals of interface-based programming.

OUTLOOK

A brief outlook to the next issue

121