PROGRAMA de Programación con Objetos 3

Carreras: Tecnicatura Universitaria en Programación Informática - Licenciatura en Informática

Asignatura: Programación con Objetos 3

Núcleo al que pertenece: Avanzado

Profesor: Javier Gelatti

Asignaturas Correlativas: Programación con Objetos 2

Objetivos:

Que las personas que cursen la materia:

- Sean capaces de utilizar tecnologías y conceptos de programación que extienden el modelo orientado a objetos clásico.
- Conozcan tecnologías y conceptos modernos que se utilizan con frecuencia en la industria.

Contenidos mínimos:

- Introducción a los sistemas de tipos y chequeo de tipos en un lenguaje de programación con objetos: tipos nominales y estructurales, tipado explícito e implícito. Duck typing. Inferencia de tipos. Esquemas de binding, early / late binding.
- Variantes del paradigma de objetos. Bloques y closures. Non-local returns. Herencia simple y múltiple; mixins y traits. Programación orientada a objetos basada en prototipos. Introducción a la programación orientada a aspectos. Open classes. Extensiones al paradigma de objetos mediante la introducción de conceptos provenientes del paradigma funcional.
- Desarrollo de aplicaciones sencillas utilizando las variantes del paradigma de objetos. Construcción de programas multilenguaje y multiparadigma. Implicancias en el diseño, patrones de diseño en las diferentes variantes del paradigma, behavioral completeness.
- Meta programación, programación reflexiva, introspección, self-modification. Mirrors.
- Lenguajes específicos de dominio (DSL). Clasificación de los DSLs: compilados, interpretados; traductores; embebidos. Creación de DSLs. Programación declarativa.

Carga horaria semanal: 4 hs

Programa analítico:

Unidad 1: Esquemas no tradicionales para definir el comportamiento de un objeto.

Bloques y closures; aprovechamiento de los bloques para la definición de estructuras de control. Programación orientada a objetos basada en prototipos. Alternativas a la

herencia simple en lenguajes basados en clases: herencia multiple, traits, mixines. Análisis comparativo de estas alternativas: problema del diamante, definición de restricciones, linearización versus flattening. Otros mecanismos para definir comportamiento: open classes, extension methods, multimethods, definiciones implícitas. Escenarios de uso y de combinación de las características y variantes mencionadas. Impacto en el diseño con objetos: behavioral completeness, simplificación de patrones de diseño, construcción de objetos a partir de la combinación de definiciones de comportamiento. Relevancia en la industria de desarrollo de software: aplicación de los conceptos introducidos en lenguajes de programación de uso extendido, como JavaScript, Scala, Java 8, Ruby, Python, etc.

Unidad 2: Metaprogramación.

Modelo y metamodelo, distintas perspectivas de los mismos objetos. Concepto de programación reflexiva. Distintas funcionalidades relacionadas a la metaprogramación. Capacidades básicas: consultas sobre clases y métodos, envío reflexivo de mensajes. Self-modification: agregado de métodos y clases en ejecución. Intercession: agregado de características a un lenguaje de programación mediante programas escritos en el mismo lenguaje. Variantes en metamodelo: mirrors, meta-objects, meta-object protocol (MOP). Relevancia de las capacidades reflexivas, uso en frameworks que proveen funcionalidades genéricas (interfaz de usuario, persistencia, distribución, etc.). Aplicabilidad de los conceptos y capacidades introducidos en distintos lenguajes de programación: formas que toma el metamodelo, distinto alcance de las capacidades reflexivas en distintos lenguajes.

Unidad 3: Sistemas de tipos y esquemas de binding.

Revisión del concepto de binding y de las consecuencias del binding dinámico. Revisión de la idea de method lookup, extensiones ligadas a las distintas formas para definir el comportamiento de un objeto. Revisión del concepto de sistema de tipos. Consecuencias del uso de sistemas de tipos en lenguajes de programación: chequeo, documentación, información aprovechable para análisis y refactors automáticos de código. Variantes y conceptos relacionados con sistemas de tipos: distintos conceptos (clases, interfaces, mixines, tipos básicos) que pueden ser considerados tipos, tipos función, generics, chequeos. Impacto del uso de tipos en capacidades reflexivas. Distintos criterios para analizar la relación de un lenguaje de programación con el concepto de tipo: amplitud y oportunidad (estático, dinámico) del chequeo; presencia o ausencia de la declaración del tipo de cada elemento incluyendo la variante que da la inferencia de tipos, distinta forma de especificación de tipo esperado (tipado nominal o estructural). Análisis y comparación de diferentes lenguajes de programación en base a los conceptos aprendidos. Aplicabilidad de los distintos enfoques, ventajas y desventajas.

Unidad 4: Lenguajes específicos de dominio

Concepto de lenguaje específico de dominio (DSL). Comparación con los lenguajes de propósito general (GPL), objetivos aplicabilidad, ventajas y desventajas. Clasificación de los DSLs: Compilados, interpretados; Traductores; Embebidos. Complejidades en la creación de un DSL: modelo semántico; gramática: concepto, relación con modelo semántico, proceso de diseño. DSL internos y externos.

Unidad 5: Programación declarativa

Revisión del concepto de programación declarativa; utilidad de los esquemas mixtos declarativo-objetos. Concepto de motor. Distintas formas de proveer declaraciones: programática, por texto con formato bien definido (p.ej. JSON), annotations. Relación con DSLs y metaprogramación. Lenguajes con características declarativas. Ventajas y desventajas de la construcción de programas multilenguaje y multiparadigma.

Bibliografía obligatoria:

- Gilad Bracha and William Cook. Mixin-based inheritance. In Proc. of the Joint ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications and the European Conference on Object-Oriented Programming, October 1990.
- Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz and Andrew P. Black, Traits: Composable Units of Behavior, Proceedings of European Conference on Object- Oriented Programming (ECOOP'03), 2743, 248–274, Springer Verlag, 2003.
- Paolo Perrotta. Metaprogramming Ruby: Program Like the Ruby Pros. Pragmatic Bookshelf, 2010.
- Martin Odersky, Lex Spoon, and Bill Venners. Programming in Scala, Second Edition. Artima, 2010.
- Chris Smith. What To Know Before Debating Type Systems. Online en http://blog.steveklabnik.com/posts/2010-07-17-what-to-know-before-debating-type-systems. 2017.
- Martin Fowler, Rebecca Parsons. Domain Specific Languages Addison-Wesley, 1st edition (23 Sep 2010).

Bibliografía de consulta:

- Gilad Bracha and David Ungar. Mirrors: Design Principles for Meta-level Facilities of Object-Oriented Programming Languages. In Proc. of the ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications, October 2004.
- Gilad Bracha. Pluggable Type Systems. OOPSLA04 Workshop on Revival of Dynamic Languages. 2004
- Gilad Bracha, Peter Ahe, Vassili Bykov, Yaron Kashai, William Maddox and Eliot Miranda. Modules as Objects in Newspeak. Proceedings of the 24th European Conference on Object Oriented Programming, Maribor, Slovenia, June 21-25 2010. Springer Verlag LNCS 2010.
- Niels Boyen, Carine Lucas, Patrick Steyaert. Generalised Mixin-based Inheritance to Support Multiple Inheritance. Technical Report vub-prog-tr-94-12, Vrije Universiteit Brussel, 1994.
- Sherman R. Alpert, Kyle Brown, and Bobby Woolf. The Design Patterns Smalltalk Companion. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 1998.
- Kiczales, Gregor; John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming.

Proceedings of the European Conference on Object-Oriented Programming, vol.1241. pp. 220–242. 1997

Organización de las clases:

Las clases se desarrollarán mediante un esquema mixto teórico-práctico, en bloques de cinco clases. La primera clase de cada bloque será fundamentalmente expositiva, pero en la que se destinará tiempo para la resolución de ejercicios. Al finalizar la misma, se entregará el enunciado del trabajo práctico. Luego, se complementarán los temas mediante una modalidad aula invertida, y se desarrollará en laboratorio.

Trabajos prácticos:

Cada trabajo práctico cuenta con una explicación previa para evacuar dudas sobre el enunciado. Los trabajos prácticos son integradores, y cuentan con un seguimiento semana a semana por parte del equipo docente.

TP1: Metaprogramación y extensiones al paradigma de objetos

Este trabajo práctico consiste en la implementación de una nueva funcionalidad en el contexto de un lenguaje de objetos, utilizando metaprogramación.

Los objetivos de este trabajo son:

- Familiarizarse con el metamodelo de un lenguaje de programación de uso extendido, y hagan un desarrollo basado en el mismo.
- Aplicar técnicas de metaprogramación para resolver un problema concreto.
- Desarrollar una aplicación sencilla utilizando herramientas no convencionales del paradigma de objetos, entendiendo sus implicancias en el diseño.
- Comprender y practicar el uso de bloques/closures en sus diferentes variantes.
- Explorar alternativas a la herencia simple en lenguajes basados en clases.

Los temas que se tratan en el trabajo práctico son:

- Mecanismos para definir comportamiento: mixins, traits, open classes y herencia múltiple. Comparación de las distintas alternativas: problema del diamante, definición de restricciones, linearización versus flattening.
- Herramientas para reificar colaboraciones: expresiones lambda, closures y full closures.
- Modelo y metamodelo, concepto de programación reflexiva, introspección, self-modification e intercesión.
- Revisión de la idea de method lookup, extensiones ligadas a las distintas formas para definir el comportamiento de un objeto
- Introducción a herramientas de lenguajes de programación orientados a objetos: multimétodos, prototipos, aspectos.

- Variantes en metamodelo: mirrors, meta-objects, meta-object protocol (MOP).

TP2: Programación objetos-funcional

Este trabajo práctico consiste en el desarrollo de un programa utilizando un lenguaje de programación estáticamente tipado con soporte para objetos y características del paradigma funcional.

Los objetivos de este trabajo son:

- Ganar experiencia en la construcción de programas combinando herramientas de los paradigmas orientado a objetos y funcional.
- Afianzar conocimientos sobre los sistemas de tipos, verificación de tipos, y distintas formas de definirlos.
- Mejorar la comprensión sobre las distintas maneras de organizar el comportamiento de los objetos, y su impacto en el diseño.

Los temas que se tratan en el trabajo práctico son:

- Construcción de programas multiparadigma.
- Sistemas y chequeos de tipos en un lenguaje orientado a objetos: consecuencias del uso de sistemas de tipos, conceptos que pueden ser considerados tipos.
- Distintos criterios para analizar la relación de un lenguaje de programación con el concepto de tipo. Tipos nominales y estructurales, tipado explícito e implícito. Duck typing. Inferencia de tipos.
- Extensiones al paradigma orientado a objetos a través de conceptos provenientes del paradigma funcional.
- Definiciones implícitas, extension methods.

TP3: Programación declarativa usando DSLs

Este trabajo práctico consiste en el desarrollo de un DSL declarativo, utilizando conceptos de metaprogramación.

Los objetivos de este trabajo son:

- Ganar experiencia en la construcción de DSLs, utilizando metaprogramación.
- Introducir el concepto de programación declarativa.

Los temas que se tratan en el trabajo práctico son:

- Concepto de lenguaje específico de dominio (DSL). Comparación con los lenguajes de propósito general (GPL), clasificación de los DSLs.

- Complejidades en la creación de un DSL.
- Concepto de programación declarativa, motor, distintas formas de proveer declaraciones. Lenguajes con características declarativas.
- Esquemas de binding, early / late binding.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidades libre y presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18).

En la modalidad de libre, se evaluarán los contenidos de la asignatura con un examen escrito, un examen oral e instancias de evaluación similares a las realizadas en la modalidad presencial.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob.	Lab.	Otros Especificar	
1	Unidad 1: Esquemas no tradicionales para definir el comportamiento de un objeto.	X	X			
2		X		X		
3	Unidad 1: Esquemas no tradicionales para definir el comportamiento de un objeto. + Unidad 2: Metaprogramación.	X	X			
4		X		X	Checkpoint TP1	
5						X
6	Unidad 3: Sistemas de tipos y esquemas de binding.	X	X			
7		X		X		
8		X	X			
9		X		X	Checkpoint TP2	
10						X
11	Unidad 4: Lenguajes específicos de dominio + Unidad 5: Programación declarativa	X	X			
12		X		X		
13		X	X			
14		X		X	Checkpoint TP3	
15						X
16	Recuperatorio					X

***INDIQUE CON UNA CRUZ LA MODALIDAD**

Firmado digitalmente por: SUAREZ Mariana Alejandra
Directora del Departamento de Ciencia y Tecnologla
UNIVERSIDAD NACIONAL DE QUILMES