Programación en Lenguaje Java

Programa de la Asignatura

Michael González Harbour

Mario Aldea Rivas

Departamento de Matemáticas,
Estadística y Computación

Presentación

Java es actualmente uno de los lenguajes de programación más populares

- el más popular según el ranking de 2014 de IEEE Spectrum
- http://spectrum.ieee.org/computing/software/top-10-programming-languages

Es un lenguaje de gran importancia en informática y en muchos de sus ámbitos de aplicación

- Numerosas empresas lo utilizan como lenguaje para el desarrollo de sus nuevas aplicaciones informáticas

Presentación (cont.)

En el primer curso del grado en Ingeniería Informática de la Universidad de Cantabria se imparte la programación en Java distribuida en dos asignaturas, que son una continuación de la otra:

- 1º cuatrimestre: "Introducción al Software"
- 2º cuatrimestre: "Métodos de Programación"

Aquí presentamos un único curso de Programación en Java entresacando los contenidos relevantes de estas dos asignaturas

El curso cuenta con:

- material de presentación
- problemas o ejercicios breves
- prácticas de laboratorio a realizar sobre el computador

Objetivos de "Programación en Java"

- Conocer y comprender las ***expresiones e instrucciones básicas*** de un lenguaje de programación imperativo
- Ser capaz de diseñar, implementar y probar ***algoritmos*** y programas sencillos en ese lenguaje
- Aplicar el ***estilo modular*** en diseño del programa
- Utilizar un ***entorno de programación***
- Aplicar los principios de ***claridad y precisión*** a la programación
- Conocer los fundamentos de la ***programación orientada a objetos***
- Ser capaz de aplicar con corrección y eficacia criterios de ***descomposición modular*** de problemas en grado suficiente como para completar la programación de su solución
- Conocer con familiaridad y ser capaz de emplear correcta y eficazmente las nociones fundamentales de la programación ***orientada a objetos***, incluyendo ***diagramas de clases***

Objetivos (cont.)

- Ser capaz de ***documentar correctamente el código*** fuente de un programa
- Saber sacar partido del uso de la ***herencia y el polimorfismo*** en un diseño modular
- Saber realizar notificación y tratamiento de errores mediante ***excepciones***
- Ser capaz de realizar entrada/salida de información sobre ***ficheros***
- Ser capaz de diseñar y aplicar ***estrategias de prueba sencillas para módulos***, y comprender su uso y su necesidad en la validación de los programas

Programa de la asignatura

1. Introducción a los lenguajes de programación

- Lenguajes de alto nivel. El proceso de compilación. El ciclo de vida del software. Concepto de algoritmo. Concepto de clase y objeto. Diagramas de clases. Estructura de un programa. Estructura de un método.

2. Datos y expresiones

- Tipos primitivos. Variables y constantes. Operadores y expresiones. Conversión de tipos. Uso de funciones matemáticas. Declaración de objetos. Strings. Composición de objetos. Atributos y métodos estáticos.

3. Estructuras algorítmicas

- Instrucción condicional. Instrucción condicional múltiple. Instrucciones de bucle. Recursión. Descripción de algoritmos mediante pseudocódigo.

4. Datos compuestos

- Arrays y tablas unidimensionales. Algoritmos de recorrido y búsqueda. Arrays multidimensionales. Tipos enumerados

Programa (cont.)

5. Entrada/salida

- Entrada/salida de texto y de caracteres. E/S de números. Gráficas. Dibujos. Menús de botones.

6. Clases, referencias y objetos.

- Concepto de clase y objeto. Creación e inicialización de objetos. Tipos primitivos, referencias y objetos. Recolector de basura. Comparación de objetos. Métodos y campos de clase (o estáticos). Anidamiento de clases.

7. Modularidad y abstracción: aspectos avanzados.

- Conceptos de modularidad y abstracción. Modificadores de acceso básicos. Paquetes. Módulos genéricos. Programación con módulos predefinidos. Documentación del código fuente.

8. Herencia y Polimorfismo.

- Herencia. Clases abstractas. Polimorfismo. La clase Object.

Programa (cont.)

9. Tratamiento de errores.

- Tratamiento de errores por paso de parámetros. Excepciones. Bloques de tratamiento excepciones. La cláusula finally. Patrones de tratamiento de excepciones. Jerarquía de las excepciones. Lanzar excepciones. Usar nuestras propias excepciones. Utilización de excepciones.

10. Entrada/salida con ficheros.

- Conceptos básicos. Flujos de datos (streams). E/S de texto. E/S de texto con formato. E/S binaria.

11. Prueba de programas.

- Verificación y validación. Pruebas del software. Caja negra: particiones de equivalencia. Herramienta JUnit.

Distribución de las clases de teoría, problemas y prácticas

Presenciales:

- 20 horas teoría + 10 horas de problemas + 30 horas de prácticas
- 10 horas de tutorías + 5 horas de evaluación

No presenciales

- Trabajo en grupo y autónomo: 75 horas (3 horas semanales)

Problemas

Problemas resueltos en casa y expuestos y debatidos durante las clases de problemas

- Se publica el problema habitualmente un martes, para hacer en casa
- Se entrega el problema en moodle hasta el lunes a medianoche
  - el sistema no admite entregas retrasadas
- Se acude a clase, cada uno en su grupo
  - se expone el problema por turnos y se debate la solución

Evaluación continuada: ***10% de la nota de la asignatura***

- nota del informe de los ejercicios con informe evaluable
  - cada problema tendrá algunos ejercicios con informe evaluable y otros que no
- exposición de un ejercicio (de informe evaluable o no) en clase
- la nota se reduce a la mitad si no se acude a clase

Prácticas

Prácticas para hacer habitualmente en una sesión de 2 horas

Evaluación de las prácticas: ***30% asignatura***

- habrá tres prácticas evaluables
  - realización de una práctica en el laboratorio y entrega de un breve informe al finalizar la sesión
- por los retrasos en la entrega de los informes de las prácticas previas a cada práctica evaluable habrá una penalización:
  - 0.5 puntos por cada entrega retrasada hasta 1 semana
  - 1 punto por cada entrega retrasada más de 1 semana
- para poder aprobar las prácticas se requiere haber entregado al menos todas las memorias de las prácticas excepto una

Clases de Teoría

Evaluación continua (***10% asignatura***):

- participación en clase mediante pequeños cuestionarios
- participación en el foro de la asignatura
- participación en las wikis

Examen final (***50% asignatura***)

- cuestiones y problemas
- se pueden usar apuntes y libros

Bibliografía:
Libros recomendados

[1] D.J. Barnes y M. Kölling, “Programación orientada a objetos usando bluej”, Prentice Hall, 2013. ISBN-13: 978-8483227916

[2] The Java Tutorials: http://docs.oracle.com/javase/tutorial/

[3] Eric J. Braude, “Ingeniería de Software: Una perspectiva orientada a objetos”. Alfaomega, 2003. ISBN: 8478975756

Bibliografía:
Libros de consulta:

[4] Java Platform, Standard Edition 7. API Specification. http://docs.oracle.com/javase/7/docs/api/

[5] How to Write Doc Comments for the Javadoc Tool. http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

[6] E. Bueno, y otros. "Algoritmos y Ejercicios resueltos en Java". Prentice Hall 2003. ISBN 84-705-4024-2

[7] Ken Arnold, James Gosling, David Holmes, “The Java Programming Language”, 4th edition. Addison-Wesley, 2005

[8] Francisco Gutiérrez, Francisco Durán, Ernesto Pimentel. “Programación Orientada a Objetos con Java”. Paraninfo, 2007

[9] J Sánchez Allende, G. Huecas, B. Fernández, P. Moreno. “Programación en Java”. 3ª edición. Me Graw Hill, 2009. ISBN: 978-84-481-6107-1

[10] J. Castro, F. Cucker, X. Messeger, A. Rubio L. Solano, B. Valles. "Curso de Programacion". McGraw- Hill, 1993

Bibliografía:
Libros de consulta (cont.)

[11] Harvey M. Deitel, Paul J. Deitel. "Java : cómo programar". 9º Ed. Pearson Educación, 2012. ISBN: 978-607-32-1150-5

[12] Ian Sommerville, “Ingeniería de software” (6ª edición). Pearson Educación de México, 2002.

[13] JUnit. http://www.junit.org/